UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: RAICES: un juego serio social para revalorizar las culturas originarias
Autores: Luciano Julián Nomdedeu
Directores: Javier F. Díaz, Laura A. Fava
Carrera: Licenciatura en Informática

Resumen

En este trabajo se exponen los aspectos más destacados del proceso de creación de Raíces, un juego serio social que sirve como didáctica para la enseñanza, y busca estimular a los niños de hoy, caracterizados por el uso naturalizado de las tecnologías, a adquirir conocimiento a través de las redes sociales. La propuesta de este juego online tiene objetivos pedagógicos escondidos, objetivos que se pretende, posibiliten a los jugadores obtener un conjunto de conocimientos sobre las culturas aborígenes en forma natural y distendida, despertando curiosidad y deseos por aprender. Se intenta generar un estado emocional que propicie procesos de aprendizaje más efectivos que los métodos tradicionales de enseñanza. Las redes sociales y el tiempo que los niños pasan en ellas, pueden ser más fructíferos si además de favorecer a la socialización, a la cooperación y a la diversión, les permite adquirir conocimiento. El juego Raíces se propone colaborar en el proceso de concientización de los niños y adolescentes sobre los valores de las culturas originarias, sus problemáticas históricas y actuales, y los derechos de estos pueblos, buscando fortalecer la concepción de la Argentina como un país pluriétnico y multicultural.

Palabras Claves

Juegos serios, juegos sociales, juegos educativos, herencia cultural, juegos web, juegos para niños, motores de juego, HTML5, JavaScript

Conclusiones

Este desarrollo ha permitido verificar, desde lo técnico, las capacidades de las tecnologías HTML5 para la creación de juegos complejos embebidos en redes sociales; y desde lo educativo, la potencialidad de los videojuegos serios web como objetos de aprendizaje. La integración de los contenidos como parte del juego y no como algo forzado, fue un factor determinante para la aceptación del juego por parte de los chicos, que pudieron aprender sobre el valor de la cultura de los pueblos originarios, sin dejar de lado la diversión.

Trabajos Realizados

- *Estudio sobre el estado del arte de los juegos serios*
- *Estudio sobre el diseño de juegos*
- *Diseño del videojuego Raíces*
- *Análisis y testeo de motores de juego existentes*
- *Implementación de un motor de juego propio*
- *Implementación de Raíces*
- *Análisis de la introducción de contenidos educativos*
- *Testeos con niños*

Trabajos Futuros

- *Incorporación de nuevos niveles relativos a otras culturas originarias*
- *Creación de un modo multi-jugador*
- *Profundización de la integración del juego con redes sociales*
- *Generación y análisis de estadísticas de uso*
- *Creación de nuevos juegos serios a partir de las técnicas desarrolladas*

Fecha de la presentación: **Julio de 2015**

Índice General

1

1. Introducción

En la actualidad, los medios de aprendizaje son muy diferentes a los de hace unas décadas. Las nuevas tecnologías han hecho posible la aparición de nuevos medios para enseñar. Entre ellos, uno de los más novedosos es el del uso de videojuegos y las redes sociales.

Si bien la creación de videojuegos educativos existe desde los inicios de los videojuegos, la evolución de los mismos es constante y día a día se encuentra con nuevos desafíos y oportunidades.

Esta tesina girará en torno a la creación de un juego serio social que buscará tornar más atractivo para los chicos el aprendizaje sobre los pueblos originarios de Argentina, de forma que a través del juego, recurso al que están habituados y conocen bien, puedan interesarse y aprender más sobre culturas que son bien nuestras.

1.1. Motivación

Las motivaciones para emprender el diseño e implementación de un videojuego serio para redes sociales son numerosas. Por un lado, se conoce la elevada cantidad de horas que los niños y adolescentes pasan conectados a Internet, compartiendo información con amigos y jugando videojuegos. Esta afirmación está basada en una reciente encuesta realizada para el Ministerio de Cultura de la Nación que reveló que el 57% de los niños argentinos forma parte de una red social y utiliza Internet para comunicarse con sus amigos, divertirse y pasar el tiempo [1]. Por otro lado, se han realizado encuestas en establecimientos educativos urbanos públicos y privados a niños de entre 9 y 12 años, observando una fuerte tendencia por parte de los niños a jugar videojuegos entre dos y tres horas diarias, con una importante presencia de los juegos de Facebook. Estas

encuestas además de indagar sobre los hábitos, usos y preferencias respecto de los juegos, contenían algunas preguntas sobre un tema en particular: *los pueblos originarios*. En su gran mayoría, se observó un desconocimiento sobre su cultura y que muchos niños los conocían, pero como habitantes del pasado.

En los últimos años, los aborígenes de la Argentina están comenzando a recuperar el lugar y el derecho que les corresponde como pueblos originarios. A partir de la reforma constitucional de 1994 se reconoce la preexistencia de estos pueblos en el país antes de la colonización española, su derecho a la tierra, identidad, educación, lengua y cultura. Sin embargo, muchas problemáticas persisten, aquí y en todo el mundo, por lo que como complemento de la legislación existente, es fundamental el aporte de la educación y la concientización de los niños y adolescentes para valorar y preservar las culturas de estos pueblos.

Los datos obtenidos a partir de las encuestas, la expansión de las tecnologías de la información y la comunicación (TICs), y el uso cotidiano de las redes sociales, motivaron el diseño de este juego serio social para que funcione como objeto de aprendizaje y posibilite a los jugadores obtener un conjunto de conocimientos y competencias en forma natural y distendida, despertando curiosidad y deseos por aprender.

La expansión viral provista por las redes sociales es otro aspecto que motiva su utilización, permitiendo que el conocimiento y los valores que se quieren transmitir se propaguen con mayor facilidad y alcancen la mayor cantidad de gente posible.

El acceso de los niños a las computadoras del programa Conectar Igualdad[1] posibilita también, el uso del juego en un entorno escolar dentro de las aulas.

1.2. Objetivos y metodología a emplear

En este trabajo se expondrán los aspectos más destacados del proceso de creación de Raíces, un juego serio social que sirve como didáctica para la enseñanza, y busca estimular a los niños de hoy, caracterizados por el uso naturalizado de las tecnologías, a adquirir conocimiento a través de las redes sociales. La propuesta de este juego online tiene objetivos pedagógicos

[1] http://www.conectarigualdad.gob.ar/

escondidos, objetivos que se pretende, posibiliten a los jugadores obtener un conjunto de conocimientos sobre las culturas aborígenes en forma natural y distendida, despertando curiosidad y deseos por aprender. Se intenta generar un estado emocional que propicie procesos de aprendizaje más efectivos que los métodos tradicionales de enseñanza. Las redes sociales y el tiempo que los niños pasan en ellas, pueden ser más fructíferos si además de favorecer a la socialización, a la cooperación y a la diversión, les permite adquirir conocimiento. El juego Raíces se propone colaborar en el proceso de concientización de los niños y adolescentes sobre los valores de las culturas originarias, sus problemáticas históricas y actuales, y los derechos de estos pueblos, buscando fortalecer la concepción de la Argentina como un país pluriétnico y multicultural.

Los juegos serios sociales tienen un gran potencial para fomentar el aprendizaje de distintas culturas y ayudar a evitar la discriminación hacia los habitantes de estos pueblos. Desde el punto de vista de los niños aborígenes, el verse reflejados en un videojuego puede favorecer a que se autoreconozcan como tales sin temor a ser discriminados.

Este desarrollo permitirá no solo poner a prueba y verificar la eficacia de un juego social como objeto de aprendizaje, sino también el potencial del uso de tecnologías estándares Web para el desarrollo de juegos complejos embebidos en redes sociales.

A lo largo de la presente tesina se analizarán los criterios que guiaron el diseño del juego Raíces, los distintos frameworks que posibilitaron su creación, los aspectos técnicos de implementación y la integración de los contenidos educativos.

Las distintas etapas del proceso de creación y técnicas aquí descriptas, podrán servir para orientar el trabajo de otros grupos que emprendan el desarrollo de juegos serios, y especialmente el de aquellos que creen juegos sociales utilizando tecnologías basadas en HTML5.

1.3. Estructura de la tesina

En el segundo capítulo de esta tesina se abordarán aspectos teóricos sobre los denominados “juegos serios”, prestando especial atención a los juegos

educativos, culturales y sociales. Dentro de este marco se analizarán definiciones, se expondrán ejemplos y se presentarán distintos puntos de vista sobre la cuestión.

En el tercer capítulo se analizarán los aspectos más importantes del diseño de juegos y las preferencias de juego de los niños, y cómo todas estas cuestiones fueron aplicadas al diseño de Raíces. Se analizarán también las características generales de los juegos para redes sociales.

El cuarto capítulo se centrará en los motores de juego: se analizará su función general, se describirán sus módulos más comunes, se expondrán criterios para la elección de un motor de juego concreto, y finalmente se presentará un análisis comparativo de distintos motores de juego HTML5.

En el quinto capítulo se describirán los aspectos relativos a la implementación del videojuego Raíces y del motor que lo sustenta, haciendo una descripción del proceso de desarrollo y un recorrido por cada una de las funcionalidades implementadas. Se abordará también el tema de la optimización del rendimiento.

En el sexto capítulo se expondrá cómo se realizó la elección del contenido educativo presente en Raíces, y de qué forma se introdujeron estos contenidos como parte del juego. Se describirán también los testeos realizados a lo largo del proceso de desarrollo y sus resultados.

En el séptimo y último capítulo de esta tesina, se expondrán las conclusiones obtenidas a partir del trabajo realizado y se presentarán posibles líneas de trabajo futuro.

2

2. Juegos Serios

2.1. Introducción

La relación que existe entre juego y educación ha sido extensamente analizada por educadores y psicólogos. Dentro de este último campo, autores como F. Froebel y J. Piaget han destacado el rol central que ocupa el juego en el aprendizaje de los niños y su desarrollo psicológico, intelectual, social y motriz.

Las ventajas que ofrece el juego como método de aprendizaje son numerosas y diversas, Granato y Lafont [2] reconocen que, entre otras cosas, la actividad del juego: genera placer, moviliza al sujeto, desarrolla la creatividad, la curiosidad y la imaginación; activa el pensamiento divergente, y favorece la comunicación, la integración y la cohesión grupal.

El aprendizaje basado en computadoras ha ido evolucionando a través de los años, en general haciendo un uso cada vez más intensivo de la interactividad. Si bien la gran mayoría de las aplicaciones educativas tienen en mayor o menor medida un componente de interactividad, no todas pueden ser consideradas como juegos. Antes de entrar en mayores detalles sobre el estado del arte de los juegos educativos en la actualidad, se analizarán algunas definiciones del término "juego", con el fin de conocer con mayor claridad los límites entre lo que es un juego y lo que no lo es.

Es interesante recuperar en primera instancia la definición dada por K. Salen y E. Zimmerman [3] quienes sostienen que: "*Un juego es un sistema en el cual jugadores se involucran en un conflicto artificial definido por reglas, que termina en un resultado cuantificable*". Esta definición tiene elementos muy importantes, como el conflicto virtual y las reglas que deben cumplirse para resolver ese conflicto. Enuncia además, que el juego debe tener un resultado cuantificable, es decir, que se puede hacer una valorización de los resultados posibles.

También es interesante recuperar la definición de J. Juul que se basa en estos aspectos, pero analiza con mayor detalle la relación del jugador con el juego, clasifica las salidas del juego como positivas o negativas, evidencia un esfuerzo por parte del jugador para alcanzar alguna cosa, y considera que un jugador tendrá diferentes estados de ánimo según si se obtiene un resultado positivo o negativo. De esta forma, la definición dada por Juul es la siguiente: "*Un juego es un sistema formal basado en reglas, que tiene un resultado variable y cuantificable, en el cual diferentes resultados son asignados a distintos valores, el jugador realiza esfuerzos con el fin de influenciar el resultado, el jugador se siente ligado al resultado, y las consecuencias de la actividad son opcionales y negociables"* [4].

2.2. Definiciones

Si bien la idea de utilizar juegos con propósitos serios existe desde hace varias décadas, la misma parece haber adquirido en los últimos años mayor relevancia y aceptación, resultando en un incremento considerable en la producción de este tipo de juegos. Esta oleada, que algunos autores tales como Djaouti, Alvarez, y otros, sitúan temporalmente a partir del año 2002 [5], coincide con la apropiación por parte de la comunidad científica del término "serious games", o su equivalente en español, "juegos serios".

Los orígenes de este término, se remontan al libro "Serious Games", escrito en el año 1970 por Clark Abt [6], un investigador que tenía entre sus metas la utilización de juegos para entrenamiento y educación. Además de establecer algunos ejemplos sobre la utilización de juegos serios en distintos entornos, Abt presenta una definición sobre este tipo de juegos: *"Los juegos pueden ser jugados seriamente o casualmente. Nosotros nos interesamos en juegos serios, en el sentido en que estos juegos poseen un propósito educacional explícito y cuidadosamente pensado, y no han sido concebidos para ser jugados principalmente como modo de entretenimiento. Esto no significa que los juegos serios no sean, o no deban ser, entretenidos."*

Esta definición dada por Abt contiene los elementos claves de lo que son los juegos serios, y no difiere notablemente de la dada por Michael & Chen en 2005 y

ampliamente utilizada en la actualidad: *"los juegos serios son aquellos que no tienen como principal objetivo el entretenimiento o la diversión"* [7].

Cuando hablamos de juegos serios, podemos hacer referencia tanto a juegos digitales como a no digitales. Dentro de este último grupo existen ejemplos de juegos con papel y lápiz para enseñar matemáticas, o el de "La nueva simulación de Alejandría", un juego serio de 1973 que todavía se utiliza en escuelas de Estados Unidos y permite enseñar sobre el funcionamiento de la política utilizando únicamente interacción humana. Más allá de algunos casos aislados, la gran mayoría de los juegos serios que se crean en la actualidad utilizan un soporte digital, y en ellos se centrará este capítulo y el resto de la tesina en general. Para designarlos, se utilizarán de manera indistinta los términos "juegos" o "videojuegos", aunque en ambos casos se estará haciendo referencia a juegos digitales.

Los juegos serios han sido utilizado históricamente con objetivos muy diversos, sirviendo a los propósitos de distintos campos tales como: salud, educación, entrenamiento, gobierno, defensa, cultura, publicidad, conciencia social, política, religión y arte. Si bien todas estas áreas estuvieron presentes desde la primera aparición de los videojuegos serios, teniendo en cuenta la cantidad de juegos creados por categoría, Djaouti destaca la preponderancia de los videojuegos educativos durante los primeros años del movimiento. Esta superioridad de los juegos educativos se ha suavizado en la actualidad, en donde el panorama se presenta más variado y con gran cantidad de juegos en todas las áreas.

En las próximas páginas, se describirán algunos juegos serios representativos del estado del arte en áreas que de una u otra manera, resultan relevantes al desarrollo del videojuego Raíces: *educación* y *herencia cultural.* También se hará referencia al estado actual de los videojuegos serios para redes sociales.

2.3. Juegos serios en educación

La búsqueda de maneras más eficientes de enseñar ha sido siempre una cuestión central para la gran mayoría de los educadores. Aunque de probada eficiencia para transmitir conocimiento, los métodos tradicionales como los libros o la disertación oral, en ciertos casos fallan al momento de captar la atención y

motivar a sus destinatarios. Así como los medios audiovisuales han sido aprovechados con fines educativos, la posibilidad de interacción que ofrecen las computadoras a través de los videojuegos también ha sido vista por algunos como un valioso instrumento para enseñar. En muchos casos, la eficacia del aprendizaje basado en juegos ha probado ser notablemente más eficiente que otras formas de aprendizaje. Estudios expuestos por Blunt [8], examinan la diferencia en los logros académicos obtenidos entre estudiantes que utilizaron videojuegos como modo de aprendizaje, y aquellos que no lo hicieron, observando en todos los casos una diferencia significativa a favor del grupo que sí hizo uso de los videojuegos.

Uno de los casos más recientes que ha adquirido notoriedad, es el de DragonBox Algebra, un videojuego infantil creado por el profesor de matemáticas noruego Jean-Baptiste Huynh en 2012, que tiene como fin enseñar álgebra de manera divertida. Lo interesante de este juego, es que los propósitos educativos se encuentran escondidos, por lo que los niños aprenden prácticamente sin darse cuenta. El juego resulta natural y divertido, y permite la apropiación por parte de sus jugadores, de diversos conceptos matemáticos como la suma, la multiplicación, la factorización en números primos, el manejo de signos, las propiedades distributivas, la factorización de polinomios, el manejo de fracciones y el uso de paréntesis.

Un estudio realizado por la universidad de Washington a más de cuatro mil estudiantes de la educación primaria, arrojó que en menos de una hora y media de juego, la gran mayoría de ellos pudo dominar todos los conocimientos de álgebra presentados en el juego DragonBox[2].

La amplia variedad de géneros de juego existentes, hace posible la creación de material educativo para diversas áreas del conocimiento. En el campo de la enseñanza de historia, por ejemplo, podemos destacar la reciente aparición de TimeMesh [9], un juego serio educativo de aventura gráfica basado en eventos históricos reales. En este juego, orientado a niños y adolescentes de entre 11 y 15 años, los estudiantes deberán adoptar el papel de un personaje que viaja en el tiempo y participa de manera directa en acontecimientos históricos relevantes a la

[2] http://www.forbes.com/sites/jordanshapiro/2013/07/01/it-only-takes-about-42-minutes-to-learn-algebra-with-video-games

historia europea. Por el momento, presenta tres escenarios: la segunda guerra mundial, la expansión marítima europea de los siglos XV y XVI, y la revolución industrial.

Una característica interesante de esta propuesta, es la posibilidad de avanzar en el juego de manera online y colaborativa, pudiendo los docentes crear grupos restringidos para su utilización en las aulas.

Otro ejemplo novedoso dentro del campo de los juegos educativos, es el de Imagana[3], un juego aún en desarrollo que tiene como fin combatir el analfabetismo. Para lograrlo, hace un uso intensivo de la narración visual y orienta al jugador en la búsqueda de distintas letras, que al ser combinadas forman palabras. Estas palabras, a su vez representan objetos que se utilizan para avanzar en el juego. Como se puede apreciar a partir de estos ejemplos, las posibilidades que ofrecen los juegos serios educativos son numerosas y no están restringidas a temáticas específicas.

2.4. Juegos serios culturales

El límite que existe entre los juegos culturales y los juegos educativos es, en muchos casos, difuso. Si bien ambos resultan en un aprendizaje por parte de los jugadores, el segundo grupo tiene como fin exponer y difundir aspectos culturales concretos, ya sea de culturas antiguas, o actuales poco conocidas o marginadas. Estos juegos, en muchos casos además de enseñar, buscan generar conciencia social sobre la importancia del respeto por la diversidad cultural. Más allá de tener particularidades propias, la gran mayoría de los juegos culturales, pueden a su vez ser considerados como juegos educativos.

Los enfoques con los que se han diseñado los juegos culturales son numerosos. En este sentido, podemos recobrar la clasificación realizada por E. F. Anderson et al. [10], en la que distingue distintos tipos de juegos culturales: prototipos y demostradores; museos virtuales; y juegos históricos comerciales. Las primeras dos categorías hacen referencia a la reconstrucción virtual de lugares históricos antiguos, y a la creación de museos virtuales interactivos respectivamente. La tercera contempla videojuegos creados con el propósito principal de ser

[3] http://www.imagana.com

entretenidos, pero que de una u otra manera, incluyen un elevado grado de fidelidad histórica, por lo que pueden ser utilizados con fines educativos. A pesar de resultar interesantes, las características lúdicas de las primeras dos categorías son en general limitadas. La inclusión de misiones y diálogos puede agregar cierto dinamismo a los recorridos, pero muchas veces no resultan del todo suficientes para capturar la atención de los jugadores. Por el contrario, los videojuegos de la tercera categoría resultan mucho más eficaces para atraer la atención de los jugadores y motivarlos para continuar jugando, a pesar de no estar diseñados específicamente para enseñar sobre las culturas que se muestran.

En la actualidad, los últimos juegos creados en el campo de los juegos serios culturales, parecen reconocer la necesidad de motivar al jugador y siguen un enfoque más parecido al de los juegos comerciales, utilizando entornos inmersivos con un alto grado de jugabilidad. Entre los juegos culturales más recientes podemos destacar: El Codex del Peregrino, Papakwaqa y Never Alone. Estos dos últimos, al estar centrados en culturas aborígenes, resultan de especial relevancia para este trabajo.

El Codex del Peregrino[4] es un proyecto que busca acercar a los jóvenes el Camino de Santiago, una ruta de peregrinación declarada "Patrimonio de la Humanidad" por la UNESCO. El videojuego, ambientado en ese entorno espacial y cultural, busca además de enseñar la música, el arte y la historia del lugar, despertar el interés de los jugadores por esta ruta y su significado, motivándolos a transitarla en persona algún día. El Codex del Peregrino está estructurado como una aventura gráfica dividida en ocho capítulos y también está disponible para plataformas móviles.

Este videojuego, aunque diseñado con propósitos serios, está concebido para competir con títulos comerciales al integrar el contenido cultural en el hilo conductor de la historia y en las mecánicas del juego.

Con una temática diferente, más cercana a la de Raíces, el videojuego Papakwaqa [11] busca enseñar acerca de la historia y cultura de la tribu Atayal de Taiwan. Este videojuego, orientado para su uso en escuelas Taiwanesas, pone énfasis en la integración espacial-temporal y la historia económica de estos pueblos, mostrando a su vez aspectos tangibles e intangibles de los componentes

[4] http://www.elcodexdelperegrino.es

de su cultura. Para esto, el género de juego elegido fue el de Construcción y Gestión (o CMS en inglés, sigla de Construction Management Simulation). Este género de juego permite explicar claramente el concepto de progreso en distintos ejes.

Para la realización de Papakwaqa, se empleó un análisis comparativo de objetivos educativos estándares y la transmisión de conocimiento a través del juego. Este análisis identificó beneficios positivos del juego en la motivación y aprendizaje de los niños con respecto a este tema, en contraposición a la enseñanza tradicional en ese país, en donde los niños reciben información de manera pasiva y deben memorizar contenidos comprimidos y diversificados, resultando en un pobre entendimiento de la profundidad del tema. En este sentido, el aprendizaje experimental ofrece numerosas ventajas.

Otro videojuego referente a la vida y cultura de poblaciones aborígenes es el caso de “Never Alone”[5], un juego de plataformas comercial lanzado recientemente para consolas de octava generación, i.e. PS4 y Xbox One.

Este juego, creado en asociación con la comunidad nativa de Alaska, está basado en la historia de “Kunuuksaayuka”, un relato acerca de una ventisca interminable que acorrala a una pequeña aldea, impidiéndoles cazar o recolectar alimentos.

Muchos elementos importantes de este juego (personajes, arte, paisajes, ambientación, etc.) se han basado y han sido inspirados por distintas historias tradicionales de Alaska, su folclore y perspectivas culturales.

Además, para profundizar la conexión personal del jugador con el pueblo nativo de Alaska y el entendimiento de su forma de vida, el juego ofrece material especial (incluyendo entrevistas con ancianos, contadores de historias y cazadores) que puede ser desbloqueado a lo largo del mismo y provee una riqueza adicional a la experiencia de juego.

Como se puede apreciar a partir de estos ejemplos, los juegos serios culturales, al igual que los juegos educativos en general, no están atados a un género de juego en particular.

2.5. Juegos serios sociales

[5] http://neveralonegame.com

Así como la aparición de las primeras videoconsolas hogareñas motivó la proliferación de los primeros juegos serios, la masificación del acceso a internet y los cambios tecnológicos de la última década han generado un nuevo estímulo para su desarrollo. La popularización reciente de las redes sociales como plataformas de juego y sociabilización, ha posibilitado el desarrollo de un nuevo estilo de juegos serios: los juegos serios sociales. Estos juegos, además de incluir elementos tradicionales, tienen un componente de interacción social. Esta interacción, que puede ser más o menos limitada, en la mayoría de los casos sirve a los fines de aumentar la cantidad de jugadores, posibilitando la expansión del juego.

El caso más popular de juego serio social existente hasta la fecha, es el caso de “Half the sky”[6], lanzado en Marzo de 2013 como parte de una plataforma transmedia que también incluye un libro ("Half the Sky: Convirtiendo Opresión en Oportunidad para mujeres de todo el mundo") y una serie de televisión.

Este juego, que cuenta con el apoyo de grandes empresas y organizaciones (Intel, Naciones Unidas, Zynga, Rockefeller Foundation, Ford Foundation, etc.) embarca a los jugadores en una aventura que apunta a generar conciencia y donaciones para ayudar a mujeres y niñas de todo el mundo. El hilo principal de la historia consiste en ponerse en el lugar de una mujer que debe ir superando pruebas y viviendo historias relacionadas con desafíos reales que mujeres y niñas deben enfrentar a diario.

[6] https://www.facebook.com/HalftheGame

Fig. 2.1: Captura de “Half The Sky Movement: The Game”

Una fuerte campaña publicitaria inicial, y las características sociales del mismo, posibilitaron que este juego ganara gran atención en poco tiempo, logrando superar el millón de jugadores y recabando donaciones por más de cuatrocientos mil dólares.

Otro videojuego serio social que ha obtenido buenos resultados, es "EnerCites"[7], un CMS lanzado en 2011 y orientado particularmente a jóvenes de Europa, en el que se puede experimentar implicaciones relacionadas a la energía (e.g. consumo, ahorro, medio ambiente, etc.).

El objetivo del juego es crear y expandir una ciudad virtual, enfrentando problemas como la contaminación, la falta de energía y las ventajas del uso de fuentes de energía renovables.

EnerCities fue publicado en Facebook, lo que ha aumentado su conocimiento y popularidad.

[7] http://www.enercities.eu/project/

Fig. 2.2: Captura del juego “EnerCities”

Además de esta estrategia de difusión, resulta interesante el trabajo realizado por sus desarrolladores para su inclusión en aulas escolares. En estos casos, además del juego, se ofreció a los docentes un documento (denominado "Teacher Toolbox") con instrucciones de instalación, propuestas de utilización didáctica y material complementario. Como resultado, miles de estudiantes de más de 110 escuelas de toda Europa, han jugado EnerCities. Muchas de ellas, además, lo han incluido como parte de su currícula.

La investigación realizada por E. Knol et.al. [12] sobre el uso de este juego, ha expuesto que los jugadores de EnerCities muestran altos niveles de concientización (e intenciones de cambiar su comportamiento) con respecto al ahorro de energía en el hogar, en comparación con el grupo de control integrado por los que no jugaron.

2.6. Discusión actual en torno a juegos serios educativos

Uno de los ejes principales de discusión actual en torno a los juegos serios educativos, se centra en el uso de este tipo de juegos en las aulas. Con respecto a esta cuestión, en “Videojuegos y el futuro de la educación” [13], I. Bogost

plantea que el uso de videojuegos en ámbitos educativos puede ser visto desde distintas perspectivas. Por un lado, rescata la opinión que la autora B. Laurel realiza desde un punto de vista crítico, quien sostiene que la inclusión de videojuegos en el aula resulta únicamente accesoria y no corrige los problemas de base que el sistema educativo actual presenta. En contraste con esta visión, Bogost sostiene que otros autores ven en el uso de videojuegos en el aula, una herramienta para tornar más interesante y eficaz el proceso de aprendizaje.

Por otra parte, también existe una creciente discusión sobre las posibilidades que ofrecen los videojuegos más allá de su uso en las aulas escolares. Entre las voces que se proclaman a favor de las capacidades educadoras de los videojuegos fuera de las aulas, W. Whright sostiene que actualmente este proceso está ocurriendo de manera invisible y online, cuando los niños vuelven a sus casas y juegan, adquiriendo habilidades y aprendiendo sin darse cuenta, sin ser forzados, siguiendo sus intereses.

En este sentido, J.P. Gee [14] analiza las ventajas que ofrecen los videojuegos comerciales, no serios, en relación con distintos tipos de aprendizaje. Destaca por ejemplo, que juegos como Pokémon pueden motivar a los niños a aprender a leer de manera mucho más eficaz que otros recursos que se les ofrece a la misma edad. De manera similar, resalta que ciertos juegos de estrategia promueven el pensamiento sistemático y el análisis de contextos amplios. Este autor identifica alrededor de treinta principios de aprendizaje que los juegos promueven, incluyendo el avanzar hacia objetivos experimentalmente en lugar de hacerlo de manera directa, la interpretación de los fracasos como desafíos, el entender las interacciones de sistemas complejos y el aprendizaje de contenido embebido en dominios de conocimiento. En este mismo sentido se orienta la investigación de la pedagoga y docente G. Esnaola [15], quien reconoce ventajas similares en el aprendizaje a partir del uso de videojuegos.

Es importante mencionar, que las capacidades educativas que analizan estos autores no son consecuencia, en la mayoría de los casos, de una decisión por parte de los creadores del juego. Estos juegos, por definición, no son serios, sin embargo posibilitan la adquisición de competencias reales y, en muchos casos, transmiten conocimientos sobre la temática abordada por el juego. Los juegos serios educativos, por el contrario, a pesar de haber sido creados con objetivos pedagógicos, en general no resultan divertidos.

Este problema ha sido reconocido por varios autores: M. Zyda [16], por ejemplo, afirma que “la industria del videojuego ya ha sido testigo del fracaso del edutainment, una combinación extraña de software educacional, levemente impregnado de diálogos tiernos e interfaces similares a las de los juegos”. Según Zyda, a pesar de que la pedagogía es un componente implícito de los juegos serios, la misma debe ser secundaria al entretenimiento, al punto de afirmar que un juego serio que no es divertido no tiene ninguna utilidad, independientemente de su contenido o valor pedagógico. De manera similar, L. Rieber [17] sostiene que: “Un juego serio efectivo, debe tener en cuenta ingredientes motivacionales intrínsecos, tales como desafío, curiosidad, fantasía y control.”

I. Bogost, fuerte promotor de la educación basada en juegos y crítico del estado actual de los juegos serios, ha expresado que la gran mayoría de los juegos serios no se preocupan lo suficiente en ser “juegos”, sino que se crean aprovechando la popularidad del movimiento, y resultan solo instrumentales. Este tipo de juegos, resultan novedosos y atractivos para la prensa y la comunidad académica, pero no alcanzan los niveles de calidad estética y de diseño (game design) requeridos para que un juego genere el deseo ser jugado una y otra vez. En este sentido, motiva a la comunidad de desarrolladores de juegos serios, a que realicen juegos que hagan justicia al medio y valgan no solo por lo que se lee sobre ellos, sino por lo que el juego ofrece realmente [18].

La creación de un videojuego serio que resulte entretenido y atrape a los jugadores es una tarea compleja. Dependiendo del ámbito en que se vaya a utilizar, esta cuestión será más, o menos importante. En ciertos ámbitos, como puede ser la capacitación profesional, la rehabilitación, o aún las aulas escolares, los requisitos de calidad no deberían ser tan estrictos. Con que los juegos ofrezcan cierta ventaja sobre las maneras tradicionales de trabajo en esos entornos, los mismos ya tendrán su utilidad. En estos casos pueden darse dos posibilidades: o los propios jugadores reconocen las ventajas de utilizar el juego para adquirir un beneficio, o el uso del juego ha sido impuesto.

En el caso de los juegos serios “no impuestos” o utilizados fuera de entornos restringidos (i.e. juegos pensados para ser jugados en momentos de ocio), la cuestión es diferente. Estos juegos deberán competir con otras fuentes de entretenimiento. En estos casos de uso “no impuesto”, los juegos deben resultar

interesantes y estar al nivel (en cuanto a entretenimiento y calidad) que ofrecen otras alternativas no serias.

En particular, para los videojuegos que buscan transmitir un mensaje y alcanzar cierta masividad, el hecho de que sean divertidos y logren atrapar a los jugadores resulta fundamental. El videojuego Raíces, amén de las posibilidades de su uso en las aulas, fue concebido principalmente como un juego de uso general, no impuesto. Los jugadores de Raíces deberían estar motivados para jugar desde sus casas, preferir este juego antes que a otro (al menos en algún momento), tener deseos de seguir jugando. Por estas razones, el tema de la diversión y su atractivo más allá de lo educativo fue siempre una cuestión central.

3

3. Diseño del juego

3.1. Introducción

Para emprender la realización de un videojuego, es necesario considerar dos cuestiones fundamentales: sobre qué tratará el juego, y qué tipo de juego será. La primera cuestión hace referencia a la temática del juego, es decir, en qué entorno estará ambientado, quiénes serán sus protagonistas, qué historia se quiere contar. La segunda cuestión, se relaciona con el género del juego y las mecánicas que el mismo presentará. Al hablar de mecánicas, se hace referencia al conjunto de acciones o formas de juego permitidas por las reglas, que guían el comportamiento del jugador, creando interacciones [19].

En algunos desarrollos, la elección del tipo de juego surgirá primero y la temática después. En estos casos, primero se tiene la estructura básica del juego, y luego, la misma es revestida por una historia o tema en particular. Esta estructura base, podrá ser ajustada para adaptarse a distintas temáticas, buscando la más adecuada según las características del proyecto en cuestión.

En el caso de los juegos educativos y culturales, ocurre lo inverso. Primero se tiene la temática que se pretende transmitir, y sobre esta temática se escoge el género de juego y las mecánicas que mejor se adapten a la transmisión del contenido. En el caso de Raíces, este último fue el esquema adoptado.

3.2. Sobre la temática del juego

La temática que abordará el videojuego serio a desarrollar, es la de los pueblos originarios de nuestro país. La enseñanza de este tema en las escuelas, ha sufrido numerosos cambios en los últimos años. Aunque positivos, según la visión de docentes y antropólogos, aún persisten ciertas concepciones o formas de

trabajo que hacen que el tema sea tratado de manera simplista y a veces contradictoria.

La antropóloga Gabriela Novaro [20], investigadora de este tema y la enseñanza de esta temática en escuelas, reconoce que en la actualidad existe la propuesta de buscar nuestras raíces en las poblaciones americanas originarias, valorarlas y respetarlas. Sin embargo, critica que “el tema tiende a abordarse como si se tratara de pueblos congelados, siempre iguales a sí mismos, omitiendo o minimizando las referencias a su conflictiva articulación con la sociedad nacional y en general a su historia”. En su investigación, identifica que la forma más frecuente de abordar el tema, consiste en que los alumnos, divididos en grupos, expongan las características de todos los grupos indígenas argentinos, y finalicen llenando tediosos cuadros sinópticos sobre “pueblo, viviendas, armas, cultura” de cada grupo. En este sentido, sostiene que “la construcción de una visión de las sociedades aborígenes como sociedades complejas, es obstaculizada por esta suerte de zapping temático caracterizado por la abundancia de información desconectada”. Como puede verse, la temática que aborda el juego es compleja.

Al desarrollar un juego serio, la elección del género o tipo de juego, debe siempre realizarse en función de los propósitos del juego, sus destinatarios y el lugar en el que será jugado.

La temática de los pueblos originarios de nuestro país es amplia y puede ser abordada desde distintos puntos de vista. En el caso de este juego, el propósito más inmediato es acercar a los niños que lo jueguen, a un tema que en general es visto como algo tedioso o aburrido.

Para evitar esta concepción, el tipo de juego a elegir, debía ser alguno al que los niños estuvieran habituados y les despertara desde un primer momento deseos de jugar. El contenido a transmitir, que involucra aspectos de la cultura, historia y problemáticas históricas y actuales de los pueblos originarios, debería poder irse desprendiendo a medida que el jugador se involucra más y más con el juego, de una manera incremental. Si la carga de información fuera demasiado elevada en un principio, los niños probablemente abandonarían el juego rápidamente.

3.3. Búsqueda de un género de juego

Con el fin de identificar los gustos y preferencias de niños de entre 9 y 12 años con respecto al uso de juegos, se elaboró una encuesta incluyendo preguntas para determinar cuáles son los juegos más jugados (en general), los juegos más jugados en Facebook, qué aspecto de los juegos resulta más atractivo, la cantidad de horas por día que dedican los niños al juego, y sus preferencias en cuanto a la dificultad de niveles, la personalización del avatar y el uso de redes sociales.

La Figura 3.1 muestra una imagen con la encuesta realizada.

Fig. 3.1: Encuesta inicial para determinar gustos y preferencias

En estas encuestas participaron más de 300 niños de escuelas públicas y privadas de la región. El análisis de las mismas, ha permitido una mejor comprensión sobre la relación actual de los niños con respecto al juego.

Como parte de este análisis, se observó una fuerte presencia de juegos con elementos del género de plataformas. En los juegos de plataformas, el jugador debe moverse de izquierda a derecha saltando sobre plataformas suspendidas en el aire, mientras supera distintos obstáculos. Con esta característica como base, la mayoría de los juegos identificados, presentaba además características propias que los alejaban en mayor o menor medida del estilo clásico asociado a este género (i.e. Mario Bros).

Dentro del género de plataformas, existen a su vez distintos subgéneros. En este sentido, una investigación realizada por Nygen et.al. [21], reconoce que dentro de los juegos de plataforma existen diferentes aspectos que pueden resultar más o menos interesantes para los jugadores. Aquí se identifican tres patrones de preferencia o subgéneros: **combat**: el jugador debe combatir enemigos, **flow**: el jugador debe moverse hábilmente por el terreno y **puzzle**: el jugador debe explorar distintos caminos y razonar.

El cuadro de la Figura 3.2 presenta los juegos de plataforma más jugados según los resultados de la encuesta realizada, identificando para cada uno sus principales componentes:

	Combat	Flow	Puzzle
Super Mario Classic	X	X	
Stick Run		X	
Transformice		X	
Fireboy and Watergirl			X
Red Ball	X	X	
Wild Ones	X		X
Icy Tower		X	

Fig. 3.2: Encuesta inicial para determinar gustos y preferencias

Además de estos resultados, se ha tenido en cuenta la investigación realizada por J. Fromme [22] sobre las distintas preferencias de juego entre niños y niñas. Con respecto a este tema, Fromme identifica que los juegos favoritos de los niños son: Acción y Combate (33%), Deportes (21%) y Plataformas (17%); mientras que en el caso de las niñas se tiene: juegos de Plataformas (48%) y juegos de Pensar (20%).

Con el fin de que Raíces contemple las preferencias de juego individuales y resulte atractivo para todos, se optó por hacer un juego de plataformas dividido en tres caminos: **Combate**, **Correr** y **Pensar**. Cada uno de estos caminos, además de tener como base los juegos de plataformas, contará con elementos y mecánicas específicas correspondientes a los subgéneros Combat, Flow y Puzzle, respectivamente.

Por otra parte, teniendo en cuenta que el juego sería embebido en la red social Facebook, se decidió incluir **características casuales** como parte del diseño de Raíces: se aprende a jugar en menos de un minuto, de fácil comprensión, compuesto por partidas breves, gratuito, re-jugable y de carga e inicio accesible y rápido.

El videojuego desarrollado se encuentra disponible en la url **http://raiceseljuego.com.ar** .

3.4. Videojuegos y emociones

Resulta interesante rescatar la investigación realizada por N. Lazzaro [23] respecto de la capacidad que tienen los juegos para generar emociones. A partir de la observación de videos, expresiones faciales, respuestas de cuestionarios y notas recogidas en sesiones de juego con distintos tipos de participantes (jugadores fuertes, jugadores casuales y no jugadores), Lazzaro llega a la conclusión de que la gente juega no tanto por los juegos en sí, sino por la experiencia que estos juegos generan. Esta “experiencia de juego” está directamente relacionada con las emociones que los jugadores experimentan.

Lazzaro ha categorizado a las emociones que surgen a partir del juego (sin considerar el aspecto narrativo), en cuatro grandes grupos: Hard Fun, Easy Fun, Serious Fun, y People Fun.

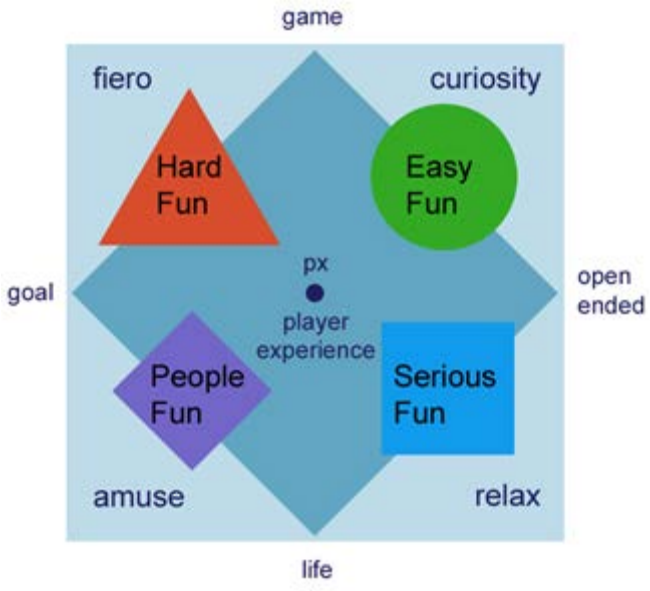


Fig. 3.3: Claves para la diversión según Lazzaro

Los juegos comerciales más populares, utilizan características de estos cuatro tipos de diversión para captar la atención del jugador y motivarlo durante el juego. A continuación se describen los tipos de diversión definidos por Lazzaro y su aplicación en el videojuego Raíces:

- **Hard Fun**: emociones provocadas por el deseo de superar obstáculos y progresar. En este tipo de diversión, la experiencia se estructura hacia la concreción de objetivos. El desafío captura la atención del jugador y genera emociones como frustración y triunfo personal, ofreciendo recompensas asociadas al progreso.

  En Raíces, esta diversión se incluye a partir de la organización del juego en niveles de creciente dificultad, ofreciendo a los jugadores visualizaciones claras sobre su progreso actual en el juego, y en comparación con otros jugadores para estimular emociones que surgen de la competencia.

  A medida que el jugador avance en el juego, su dominio del mismo crecerá, generando sensaciones de satisfacción al alcanzar logros cuando se ponen a prueba sus habilidades.

- **Easy Fun**: las emociones que surgen de esta diversión, están asociadas a la inmersión que el jugador experimenta durante el juego. Entre ellas se cuentan el misterio, la sorpresa y el asombro. En el caso de esta diversión, el tener un objetivo claro que superar, no resulta lo más atrayente. En su lugar, la estimulación visual y sonora, y el nivel de detalle ocupan un rol central. Aquí se busca crear un mundo que estimule la *curiosidad*, el querer saber que viene a continuación, el deseo de explorar.

  Para la creación del videojuego Raíces, este tipo de diversión ha sido estimulada a partir de una construcción visual y sonora que busca generar un ambiente inmersivo con gran cantidad de detalle. Para esto, se aprovechan recursos visuales como el manejo dinámico de cámara (con distintos niveles de zoom) y la organización de fondos en capas a distintas profundidades, técnica conocida como “parallax scrolling” o paralaje. Las interacciones que surgen a partir de ciertos objetos con físicas realistas, también generan efectos visuales interesantes.

  Para motivar la curiosidad y generar emociones de asombro y sorpresa, constantemente durante el avance en el juego, se introducen elementos o

personajes nuevos, cuyo comportamiento debe ser explorado por el jugador.

- **Serious Fun:** emociones que surgen cuando se pretende que jugar tenga un sentido. En esta diversión, los jugadores juegan con el propósito de pasar de un estado mental a otro (por ejemplo: relajarse, cambiar el humor), para generar algún cambio en el mundo real, o para obtener algún beneficio personal. El juego puede ser visto como un modo de terapia o como una forma de hacer que tareas aburridas se tornen interesantes.
  Las características serias de Raíces, explotan de manera natural este tipo de diversión. Aunque no ocurrirá en todos los casos, muchos niños pueden llegar a apreciar el hecho de estar aprendiendo mediante el juego y sentir emociones positivas vinculadas con este aprendizaje y el jugar de manera seria.

- **People Fun:** las emociones son provocadas a partir de la interacción con otros jugadores. Esta interacción, que puede ser *in situ* o de manera online, posibilita la aparición de emociones asociadas a actividades de competencia y cooperación.
  Al jugar con otros, las emociones emergen más frecuentemente y de manera más intensa que jugando aisladamente. Los jugadores que disfrutan de esta diversión, ven a los juegos como un mecanismo de interacción social. Raíces, a pesar de no incluir por el momento un modo multijugador, debido a su integración con Facebook, podría permitir que cada jugador reconozca el grupo de amigos que está jugando, y visualice el progreso de los mismos, estimulando la competencia y la cooperación.

3.5. Los juegos sociales

Las redes sociales pueden estimular el aprendizaje colaborativo porque facilitan la formación de grupos efectivos y afectivos, permiten la comunicación dentro de los grupos y ayudan a fortalecer las identidades individuales y colectivas.
Para introducir características sociales al diseño del videojuego Raíces, se ha tenido en cuenta la investigación de A. Jarvinen [24] sobre principios de diseño de juego, presentes en los juegos más populares de la red social Facebook. Este

autor, identifica cinco características que provocan diversión en los juegos embebidos en redes sociales: Fisicalidad Simbólica, Espontaneidad, Sociabilidad inherente, Narrativa y Asincronismo.

Las motivaciones que ofrecen el uso de las redes sociales, debería poder extrapolarse, al menos en parte, al diseño del juego. En este sentido, es importante tener en cuenta la naturaleza fugaz de cómo las personas utilizan estas redes sociales: múltiples sesiones diarias y cortas. A medida que esta forma de interacción se consolidaba, los juegos han tenido que adaptarse a las rutinas de los jugadores y no al revés.

Por otra parte, los jugadores eligen juegos sociales para convertirse en más sociales. El sentido físico inmediato de la presencia de otros jugadores conocidos, aunque en forma de avatares, es uno de los factores que favorece a la creación de un fuerte sentido de inmediatez social. Esto acompañado por la reciprocidad inmediata que se logra con este tipo de interacción, favorecen a la retención del jugador en el juego.

En la investigación de G. Esnaola [25] sobre juegos embebidos en redes sociales, se identifican ocho características distintivas de este tipo de juegos:

- Es un juego online. Necesitamos una conexión a internet para poder acceder e interactuar con él.
- Se distribuye a través de plataformas de gestión de redes sociales.
- Se inicia individualmente, pero es fundamental tener amigos o asociar a otros jugadores para poder progresar. Por lo tanto, puede denominarse multijugador.
- El juego es asíncrono.
- No terminan nunca. No hay “un ganador”, pero existe una competitividad encubierta, a través de los niveles o puntuación conseguidos en relación con otras personas (necesidad de estatus social) a las que se está atado emocionalmente.
- Generan comunidad.
- Poseen un sistema de moneda virtual.
- Poseen un sistema de reclutamiento.

El hecho de integrar como parte del juego la posibilidad de invitar nuevos amigos y de compartir contenido del avance propio de cada jugador dentro del mismo, aunque funcionalidades estándares, sirven también para fortalecer el sentido social del juego y estimular la interacción entre jugadores. Estas funcionalidades, como sostiene Brathwaite [26], sirven además para otro fin: estimular la proliferación del juego y su conocimiento a través de la red social, para que el número de jugadores aumente.

3.6. Framework MDA

Cuando se diseña un juego, se piensa inevitablemente en los jugadores, en cómo jugarán, cómo interactuarán, como se divertirán, cómo consumirán ese juego. El framework **MDA** (Mechanics / Dynamics / Aesthetics) propuesto por Hunicke et.al. [27], es una herramienta que ayuda a comprender las partes involucradas en el diseño de un juego. Este framework divide el diseño en tres aspectos:

- **Mecánicas***:* describe los componentes principales del juego a nivel de representación de datos y algoritmos. Este aspecto se asocia a las “reglas del juego”.
- **Dinámicas**: describe los comportamientos que ocurren al momento de jugar, basados en las mecánicas y dependientes de las acciones efectuadas por el usuario y su respuesta a lo largo del tiempo. Este aspecto se refiere al “sistema de interacciones” que se crea al momento de jugar un juego.
- **Estéticas**: Describe las emociones que se espera evocar en el jugador cuando el mismo interactúa con el juego. Este último aspecto se asocia al concepto de “diversión”.

El diseño de la experiencia de juego utilizando el enfoque formal MDA, consiste en un proceso iterativo que tiene su base en reconocer a los juegos como sistemas dinámicos. Estas tres capas de abstracción permiten conceptualizar la forma en que se interactúa con los juegos y a partir de este análisis comprender qué cambios realizar para obtener los resultados deseados. Estos cambios,

siempre serán cambios en las mecánicas del juego, que es el único aspecto que puede ser directamente modificado y controlado. Las mecánicas (o reglas), al ser jugadas generan dinámicas, y estas dinámicas son las que a su vez hacen que las emociones emerjan.

Tomando como marco de referencia el modelo propuesto por MDA, a continuación se describen las mecánicas, dinámicas y estéticas del videojuego Raíces.

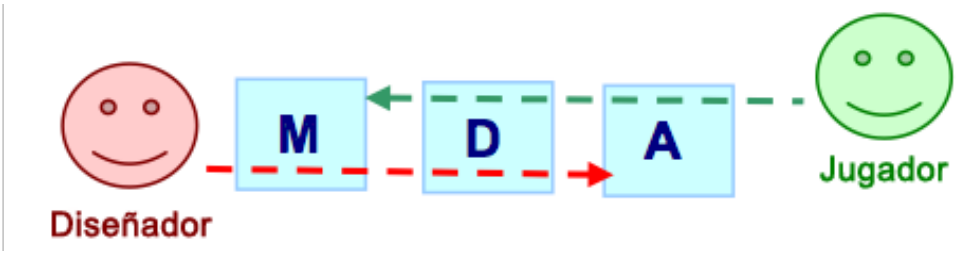


Fig. 3.4: Direccionalidad del diseño MDA

3.6.1. Mecánicas

Como se ha mencionado anteriormente, la mecánica básica de Raíces, será la de los juegos de plataforma clásicos. A su vez, los niveles presentarán características casuales y estarán divididos en tres subgéneros o caminos.

Todos los caminos tendrán una base en común: para completar cada nivel, se debe llegar a una "puerta de fin de nivel", habiendo juntado una cantidad determinada de "piezas de nivel" con importante valor simbólico.

Se describen las características básicas de cada camino:

- **Combate**: basado en luchas contra varios enemigos para pasar de nivel. Estos enemigos son criaturas que dificultan el avance del jugador y se deben ir eliminando hasta llegar al final del nivel. Los enemigos en este juego representan a personajes que han sido hostiles, a lo largo de la historia, con los pueblos originarios. En los distintos niveles se suceden en línea histórica diferentes personajes como colonizadores españoles, terratenientes, soldados, maquinarias taladoras, etc.

  La mecánica propia de este subgénero será la capacidad del jugador de lanzar flechas y saltar sobre enemigos generando distintos comportamientos.

- **Correr**: el jugador debe moverse hábilmente, saltar obstáculos, etc. En este subgénero los patrones rítmicos de los saltos y movimientos ocupan un lugar central. Los niveles presentan más precipicios, mayor velocidad de movimientos, elementos que persiguen al personaje o caen desde el cielo, restricciones de tiempo, etc. El objetivo simbólico del correr en el juego es llegar antes de determinado tiempo para poder conseguir retener aspectos de la cultura originaria, llegar antes de que se pierdan sus lenguas, llegar antes de que se extingan sus ceremonias, su música, etc.

  En estos niveles, la mecánica distintiva será la inclusión de portales que permiten un aumento creciente en la velocidad de movimiento del protagonista.

- **Pensar**: el jugador debe observar detenidamente el nivel identificando qué objetos y caminos existen, y reflexionar. Las mecánicas básicas serán: hacer crecer un árbol (todas las veces que se quiera) para llegar a lugares inaccesibles, guiar animales hacia las piezas, y la posibilidad de activar (o desactivar) plataformas móviles. Estas mecánicas deberán combinarse correctamente para alcanzar todas las piezas de cada nivel.

  Desde este subgénero se buscará abarcar aspectos relacionados a la mitología y cosmovisión de cada pueblo.

Todas estas mecánicas, estarán enriquecidas a su vez por mecánicas derivadas de la interacción social: invitar nuevos amigos, observar el progreso de otros jugadores y compartir contenido.

3.6.2. Dinámicas

Hunicke et.al. reconoce que una de las principales diferencias entre los juegos y otras formas de entretenimiento, radica en su relativa imprevisibilidad. No puede saberse, antes de que un juego sea jugado, cómo reaccionarán los jugadores frente a las mecánicas dadas.

Las dinámicas que se generan dependen de las características particulares de cada nivel y del jugador que juega esos niveles. Enumerar aquí las dinámicas surgidas en cada momento del juego, no tiene demasiado sentido. Es importante en cambio, destacar el papel que el análisis de las dinámicas (observadas en las

múltiples pruebas del juego) tuvo en el proceso de diseño de cada nivel. Luego de realizadas varias pruebas de prototipos, se identificó la aparición de dinámicas no deseadas o deficientes, que han motivado ajustes en las mecánicas. Estos ajustes, propiciaron la aparición de nuevas dinámicas, más apropiadas para la generación de las emociones buscadas.
Más allá de las dinámicas particulares de cada nivel, se han analizado también algunas dinámicas generales:

- Interés del jugador con respecto al sistema de puntos
- Uso de diálogos
- Relación del jugador con la opción de elegir uno o varios caminos
- Actitud del jugador en relación con los demás jugadores (dinámicas sociales)

3.6.3. Estéticas

El aspecto de "Estéticas" del framework MDA, al hacer referencia a las emociones que provocan los videojuegos, tiene una relación directa con las emociones analizadas en los cuatro tipos de diversión propuestos por N. Lazzaro.
El modelo MDA incluye las siguientes "Estéticas":

- **Sensación** (El juego como sentido-placer)
- **Fantasía** (El juego como "hacer creer")
- **Narrativa** (El juego como narración de conflictos / drama)
- **Desafío** (El juego como carrera de obstáculos)
- **Compañerismo** (El juego como framework social)
- **Descubrimiento** (El juego como territorio desconocido)
- **Expresión** (El juego como forma de autodescubrimiento)
- **Sumisión** (El juego como pasatiempo)

Si bien entre el modelo de emociones previamente presentado y este último existen ciertas variaciones en los nombres y la clasificación, el concepto es esencialmente el mismo, y su aplicación en Raíces ya ha sido analizada.
Las emociones de **Sensación**, **Fantasía** y **Descubrimiento** pueden asociarse al tipo **Easy Fun** expuesto anteriormente, mientras que la emoción de **Desafío** remite al concepto de **Hard Fun**. De la misma manera, puede establecerse una

relación entre **Compañerismo** y **Social Fun**, y entre las emociones de **Sumisión** y **Expresión** con la idea de **Serious Fun**.

A diferencia del análisis de las emociones realizado por N. Lazzaro, en este framework, la narrativa sí es tenida en cuenta como un elemento más en el diseño de la diversión.

4

4. Motores de juego

4.1. Introducción

Definidas algunas cuestiones básicas sobre el diseño del juego, al momento de emprender la implementación de Raíces, una de las cuestiones centrales fue la elección de un **motor de juego** (o game engine) que sirviera como base para su programación.

En términos generales, un **motor de juego** es un sistema de software encargado de gestionar y abstraer aspectos técnicos comunes al desarrollo de juegos.

Cabe aclarar que existe una clara distinción entre lo que es el código del motor de juego, y el código del juego en sí, tal como describe Lewis y Jacobson en su propia definición de motor de juego. En [28], establecen que los motores de juego son una “colección de módulos de código de simulación, que no especifican directamente el comportamiento del juego (lógica del juego) ni el entorno del juego (información de niveles)”.

La definición de motor de juego (y sus alcances) aún no es del todo clara en la comunidad científica [29]. Una de las cuestiones en discusión, es si cierto conjunto de herramientas que se utilizan para construir el contenido del juego deben ser tomadas como parte del motor. Más allá de este interrogante, A. Thorn [30] destaca la utilización de estas herramientas, que sirven como nexo entre dos extremos y logran conectar el contenido pasivo del juego con el poder activo del motor. Estas herramientas pueden ser: editores de niveles, generadores de mapas, exportadores de imágenes, editores de caminos, etc.

4.2. Módulos comunes a los motores de juego

A pesar de que cada motor tiene características propias, existen módulos generales presentes en todos ellos. Cada módulo o subsistema se encarga de algún aspecto del motor, agrupando de esta manera su funcionalidad. Además, esto permite tener el código del motor distribuido en varios archivos facilitando la corrección de errores y su modificación.
A continuación, siguiendo las líneas expuestas por A. Thorn, se describirán los módulos que tradicionalmente presenta un motor de juego, y sus tareas más importantes:

4.2.1. Administrador de recursos

Para que el juego adquiera una forma concreta, es necesaria la utilización de un conjunto de recursos digitales. En la industria del desarrollo de videojuegos, es común la utilización del término “asset digital” o simplemente “asset”. Van Niekerk nos acerca una definición de este concepto [31]: “Un asset digital es un ítem de texto o de algún otro medio, que ha sido formateado a una fuente binaria, y del cual se tienen los derechos para ser usado”.
En un videojuego podemos encontrar “assets” de distintos tipos: gráficos, animaciones, sonidos, música, archivos con información de niveles, modelos 3D, archivos de diálogos, videos, etc.
Resulta difícil imaginar un juego que no haga uso de al menos uno de estos tipos de recursos, por lo que tener un componente que los administre es deseable en cualquier motor de juego.
Las tareas de este módulo pueden resumirse en: a) identificar y distinguir entre todos los recursos disponibles en el juego, b) cargar los archivos en memoria (y descargarlos de la misma), c) asegurarse de que solo resida en memoria una instancia de cada recurso al mismo tiempo.

4.2.2. Administrador de renderizado

Aunque gracias al administrador de recursos, las imágenes y videos quedan disponibles para ser usados por el juego, es necesaria la presencia de un proceso adicional que dibuje los gráficos en pantalla mediante el hardware gráfico. Dado que prácticamente todos los juegos presentan algún tipo de gráficos, ya sea 2D o

3D, un administrador de renderizado es un componente esencial en los motores de juego. Sus tareas incluyen: a) comunicar de manera eficiente el motor de juego con el hardware gráfico del sistema, b) renderizar en pantalla las imágenes guardadas en memoria, c) establecer la resolución del juego.

4.2.3. Administrador de entrada de usuario

Como se ha mencionado anteriormente, los videojuegos se diferencian de otras formas de entretenimiento esencialmente en que son interactivos. La forma en que interactúa el usuario con el juego dependerá de los dispositivos de entrada que tenga el sistema en donde se ejecuta, pudiendo ser por ejemplo: teclado, mouse, joystick, pantalla táctil, keypad, dispositivos de detección de movimiento, etc.

La necesidad de contar con un mecanismo para recibir y responder a la entrada del usuario pudiendo lograr cierto grado de abstracción del dispositivo de entrada, lleva a los motores de juego a implementar un administrador de entrada de usuario. Su propósito es: a) leer la entrada del usuario en tiempo de ejecución, considerando todos los dispositivos aceptados por el juego, b) codificar esa entrada a una forma general, independiente del dispositivo.

4.2.4. Administrador de audio

Así como el administrador de renderizado se encarga de mostrar los gráficos en pantalla, el administrador de audio cumple una tarea equivalente al reproducir el audio del juego. La mayoría de los juegos presenta la necesidad de reproducir audio, entendiendo por audio tanto los sonidos como la música.

Sus tareas son: a) servir como nexo entre el juego y el hardware de audio, b) definir el volumen de todo el audio que aparece en el juego, c) aplicar efectos de sonido tales como efectos fade, de paneo y de eco.

4.2.5. Módulo de manejo de errores

Es importante que al momento de desarrollar un juego, el motor brinde facilidades y herramientas para la detección y corrección de errores. Dado que los errores

pueden existir tanto en el motor como en el juego, sin importar cuán rigurosamente ambos hayan sido testeados, el motor debe proveer mecanismos para tratar y brindar detalles sobre estos errores cada vez que ocurran.
Las tareas de este módulo son: a) detectar excepciones en tiempo de ejecución, b) manejar esas excepciones y en caso de ser irreparables, alertar al usuario de la ocurrencia del error, c) loguear la ocurrencia de estos errores para (posiblemente) enviar un reporte al desarrollador.

4.2.6. Administrador de escenas

El concepto de escena en un videojuego hace referencia a un conjunto de elementos que, situados en un mismo lugar (o escenario), forman un todo. Esta composición de elementos tiene un sentido para el jugador y está fuertemente relacionado a las acciones y eventos que ocurren en el juego. Por ejemplo, dado un videojuego en el que existan niveles, cada nivel puede constituirse como una escena propia.
El propósito del administrador de escenas es coordinar los distintos recursos del motor de acuerdo a lo requerido por la lógica del juego, cargando y actualizando las distintas escenas.
Entre las tareas que realiza se cuentan: a) comunicarse con los módulos de renderizado, de audio y de recursos, según las necesidades de la escena, b) mantener un registro del tiempo transcurrido, con el propósito de lanzar eventos y coordinar el movimiento de objetos, c) enumerar todos los elementos que forman parte de la escena, permitiendo al desarrollador recorrerlos y accederlos individualmente.

4.2.7. Motor de física

El administrador o motor de física, es un componente dedicado a aplicar las leyes físicas a los objetos del juego. Entre otras cosas, es el responsable de manejar los efectos de gravedad e inercia, de permitir aplicarles a los objetos fuerzas y torsiones, y de controlar su velocidad.
Es importante mencionar que no para todos los juegos será necesario contar con un motor de física. Por poner ejemplos clásicos, para implementar videojuegos

como el Tetris o el Pac-man, no sería necesario contar con un motor de física, ya que solo presentan movimientos simples que no están atados a las leyes físicas. Por el contrario, un videojuego de simulación de vuelo o de carreras de autos, deberá contar con un motor de física para obtener resultados realistas.
Sus principales tareas son: a) controlar en tiempo de ejecución el comportamiento de los objetos del juego con respecto a las leyes físicas, b) permitir identificar y manejar colisiones entre objetos.

4.3. Elección de un motor de juego para Raíces

Si bien la mayoría de los motores permite crear cualquier tipo de juegos, algunos están mejor preparados y/o presentan facilidades para ciertos géneros de juego. Por esta razón, antes de elegir un motor en particular, es importante tener definido de que género será el juego.
En vista a estas consideraciones y formalizada la idea de crear un juego de plataformas 2D, se procedió a buscar motores que contaran con las siguientes características:

Características excluyentes:

- Funcionamiento en navegadores (JavaScript + HTML5, sin plugins)
- Manejo de gráficos y tiras de imágenes
- Manejo de física básica y colisiones
- Manejo de sonido
- Manejo de recursos
- Manejo de interacción con teclado y mouse
- Funcionamiento en distintas configuraciones de hardware

Características deseables:

- Manejo de animaciones
- Zooms dentro del juego
- Portabilidad a dispositivos móviles
- Manejo de física compleja

- Editor de niveles
- Multijugador
- Flexible, para agregar funcionalidad en caso de ser necesario

La oferta de motores de juego que cumple con estas características es amplia. En el sitio [32], se listan más de setenta motores de juego JavaScript. Otro listado, no tan extenso pero que agrupa los motores de juego HTML5 más reconocidos, presenta más de veinte opciones distintas [33]. Esta variedad demuestra la actual expansión que tiene el desarrollo de juegos en HTML5.

Entre los motores más reconocidos se encuentran: Construct 2, Impact JS, Lime JS, Easel JS, Crafty JS, Isogenic Engine, Turbulenz y Phaser.

La mayoría de estos motores, presentan diferencias importantes entre ellos. Algunos proveen un gran nivel de abstracción y permiten crear juegos de manera sencilla, mientras que otros son más complejos, con menor grado de abstracción, pero a la vez más flexibles. Otra diferencia es el tipo de licencia que presentan: los hay con licencias comerciales, gratuitas y de código libre.

Con el fin de examinar distintas alternativas, se analizarán cuatro motores JavaScript/HTML5 que, aunque todos adecuados para implementar un juego de plataformas 2D, presentan cada uno distintas características y funcionalidades. Los motores sobre los que se realizará el análisis son: Construct 2[8], Impact JS[9], Playcraft[10] y Turbulenz[11].

La elección de estos motores dentro de la gran variedad existente en el mercado, fue motivada, durante el proceso de desarrollo de Raíces, por diversas cuestiones: Construct2 e Impact.js se eligieron debido a su gran popularidad y aceptación por parte de los desarrolladores de juegos HTML5; por otra parte, la elección de Playcraft y Turbulenz fue motivada por las posibilidades que ambos ofrecen para el desarrollo de juegos complejos, a pesar de ser opciones más nuevas y menos probadas por la comunidad. Estos dos últimos, a diferencia de los primeros, incluyen licencias menos restrictivas y no comerciales.

Para sentar las bases de una comparación entre los distintos motores, se eligieron un conjunto de aspectos a analizar. Este análisis, permitirá distinguir las

[8] https://www.scirra.com/
[9] http://impactjs.com/
[10] http://playcraftlabs.com/
[11] http://biz.turbulenz.com/developers

características esenciales de cada motor, y servirá como referencia a la hora de elegir un motor de juego. Adicionalmente, se elaborará una conclusión para cada motor. Según la complejidad del juego a implementar, la experiencia de los desarrolladores o el tiempo que se disponga para completar el mismo, las distintas opciones resultarán más o menos adecuadas.

El análisis surge a partir del trabajo directo con los mismos. Distintas versiones del videojuego **Raíces** han sido implementadas con cada motor, permitiendo adquirir un conocimiento en profundidad sobre la dinámica de trabajo con cada uno de ellos y sus particularidades. Los criterios elegidos para la evaluación son los siguientes:

- **Nivel de abstracción / Facilidad de uso**

  Este criterio servirá para analizar el grado de abstracción que presenta un motor a la hora de desarrollar juegos. En general, cuanto mayor es el nivel de abstracción que ofrece un motor, mayor será su facilidad de uso. Se analizará de qué manera se ocultan aspectos técnicos y comportamientos comunes a distintos tipos de juegos.

- **Funcionalidad**

  La funcionalidad de un motor estará dada por la cantidad de características que el motor abstrae e implementa de manera nativa. Dado que todos los motores incluyen manejo de gráficos, sonidos, recursos e interacción con el usuario, en este apartado se obviarán estas funcionalidades generales.

  Las características específicas que incluya cada motor, servirán para darle atractivo gráfico al juego o para permitir la creación de comportamientos más complejos entre los objetos que participan. También pueden incluir abstracciones en la comunicación con servidores, integración con plataformas externas, aspectos multijugador, posicionamiento de interfaz de usuario, etc.

- **Herramientas para creación de contenido**

  Este ítem se centrará en las herramientas que ofrece cada motor para facilitar la creación de niveles, animaciones, diálogos, imágenes, y cualquier otro contenido que da forma concreta al juego, más allá de la programación.

- **Flexibilidad**

  Si bien los motores de juego generalmente encapsulan su funcionamiento interno y la programación se realiza a través de la interfaz que los mismos exponen, en muchos casos será necesario cruzar estos límites para agregar funcionalidad o modificar la ya existente. Este apartado analizará la capacidad que tiene cada motor para ser modificado y extendido.

- **Portabilidad**

  Una característica propia de los juegos desarrollados con HTML5 es su alta portabilidad. Un mismo código puede ser ejecutado por distintos navegadores en distintos dispositivos. Si bien este soporte existe como característica de la tecnología, existen ciertas cuestiones a tener en cuenta: resoluciones de pantalla, manejo de la entrada de usuario y eficiencia. Para que el juego funcione correctamente en dispositivos móviles, algún tipo de soporte por parte del motor será requerido.

- **Licencias**

  En este apartado, se detallará si el motor se ofrece de manera gratuita o comercial, y los alcances generales de sus licencias de uso y distribución.

4.4. Análisis de distintos motores de juego

4.4.1. Construct2

Construct2 es un motor de juego 2D y generador de contenido multimedia, que permite crear juegos desde la propia interfaz del programa, sin necesidad de escribir líneas de código. Aunque está orientado principalmente a no programadores, puede ser utilizado como herramienta de prototipado rápido ya que provee implementaciones de las mecánicas de juego más comunes y permite manejar los recursos gráficos de manera sencilla.
La programación del juego se puede realizar íntegramente desde un editor visual arrastrando y soltando bloques que representan **eventos** y **acciones**. Por otra parte, las entidades del juego tienen asociados **comportamientos** y **atributos**.

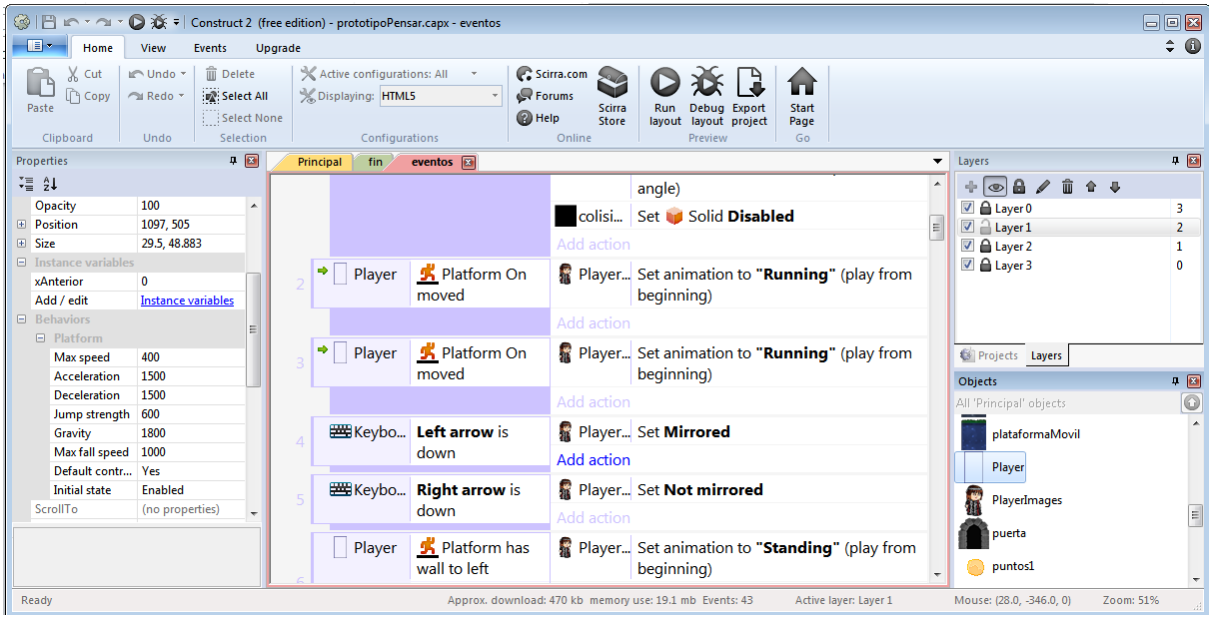


Fig 4.1: Manejo de eventos en Construct2

Esta forma de trabajo permite crear prototipos de manera rápida, permitiendo por ejemplo probar si una mecánica es entretenida, o si resulta funcional a los fines educativos planteados. Contar con un prototipo jugable en una primera etapa del trabajo resulta valioso. El testeo del mismo permite identificar las falencias y virtudes del diseño propuesto, posibilitando realizar cambios sin que los mismos signifiquen una reestructuración importante del código, como podría suceder cuando se trabaja sobre una versión avanzada del juego.

Aunque simple, a través del mecanismo de programación presentado por Construct2 pueden implementarse mecánicas complejas. Toda la lógica del juego se estructura en planillas de eventos. Cada planilla de eventos tiene una lista de eventos, que contienen sentencias condicionales, o disparadores. Cada vez que se activa un evento, se ejecutan ciertas acciones o funciones. Los eventos pueden organizarse en grupos, permitiendo activar y desactivar varios eventos a la vez, y facilitando la organización de proyectos grandes. El manejo de eventos de Construct2 permite también utilizar lógica proposicional, sub-eventos, variables locales y funciones recursivas.

Junto con el manejo de eventos, otro componente fundamental de Construct2 para definir la programación de un juego es el módulo de **Comportamientos**. Los **comportamientos** funcionan como funciones pre-empaquetadas que pueden

asignarse a los objetos y reusarse siempre que se necesiten. Al agregar un comportamiento a un objeto, este adquiere instantáneamente un conjunto de características y funcionalidad que no es necesario programar. Por ejemplo, al agregar el comportamiento “plataforma” a un objeto, inmediatamente ese objeto puede correr y saltar sobre objetos marcados como “sólidos”. A su vez, cada comportamiento tiene un conjunto de variables que permiten especificar exactamente cómo se debe comportar. Siguiendo el ejemplo del comportamiento “plataforma”, estas variables permiten especificar velocidad, aceleración, fuerza de salto, gravedad, etc.

Nivel de abstracción / Facilidad de uso

Este motor está en gran medida orientado a no programadores y principiantes, por lo que el nivel de abstracción que ofrece es muy alto. Construct2 busca posibilitar la creación de juegos de manera rápida y simple, y para lograr esto, oculta y encapsula los aspectos técnicos tanto como sea posible. En este sentido el resultado es satisfactorio. Al crearse juegos con esta herramienta, toda la programación consiste en asignar comportamientos, identificar eventos y definir acciones asociadas a esos eventos. Ningún aspecto técnico queda expuesto y solo se programa la lógica del juego en cuestión.
El manejo de imágenes y posicionamiento de entidades en cada escena también resulta sencillo e intuitivo.

Funcionalidad

Entre las funcionalidades generales que ofrece este motor podemos destacar:

- Comportamientos comunes ya implementados.
- Manejo de animaciones
- Manejo de física compleja (a través de Box2D)
- Posibilidad de crear mapas de colisiones basados en polígonos
- Visualización de colisiones en modo depuración
- Implementación de elementos de interfaz de usuario
- Abstracción en la comunicación con el servidor y con otros dispositivos

- Plugins creados por la comunidad
- Renderizado eficiente (WebGL/Canvas)

Además, el motor ofrece un conjunto de características y funciones orientadas a gráficos, entre las que se cuentan:

- Parallax scrolling (movimiento de capas a distintas velocidades)
- Efectos gráficos WebGL (blend, color y distorsión)
- Fondos en mosaico (Tiles)
- Manejo de cámara y zoom
- Sistema de partículas

Herramientas para creación de contenido

Una fortaleza de este software es su integración con herramientas de creación de contenido. Construct2 es a la vez un motor de juego y un creador de contenido multimedia, posibilitando la creación de la totalidad de un juego (lógica y contenido) sin salir del programa. Desde la misma aplicación se puede incorporar sonido, importar y editar imágenes, y establecer animaciones a partir de estas imágenes.
El editor principal provee una interfaz en la que se puede ver exactamente cómo será cada nivel, mostrando los gráficos de cada objeto en sus respectivas posiciones. Desde este editor, se puede arrastrar, rotar y escalar objetos, visualizar efectos aplicados y cambiar propiedades. También permite una organización en capas, facilitando la edición de los niveles y el agrupamiento de objetos.

Flexibilidad

Si se necesita algún tipo de funcionalidad que no se encuentre comprendida por defecto en Construct2, el motor permite desarrollar plugins y comportamientos usando un SDK JavaScript. Este SDK permite escribir código JavaScript e integrarlo a los juegos que se desarrollan con Construct2, brindando una alternativa a la programación a través de la interfaz de usuario. Los plugins

permiten también crear efectos visuales propios a través del lenguaje de sombreado GLSL.
Aunque la creación de plugins agrega flexibilidad, el funcionamiento interno del motor siempre permanece oculto.

Portabilidad

Construct2 permite exportar a distintas plataformas HTML5, soportando los navegadores Google Chrome, Firefox, Internet Explorer 9+ y Opera en computadoras de escritorio. Si el juego se ejecuta en un navegador que no soporte WebGL, se activará automáticamente un modo de compatibilidad utilizando el elemento Canvas de HTML5.
En cuanto a portabilidad en móviles, el programa incluye soporte nativo para exportar a iOS y Android utilizando CocoonJS, appMobi y PhoneGap.
Los juegos se exportan desde la aplicación simplemente seleccionando la plataforma destino deseada, aunque a veces se requieren ajustes adicionales.

Licencia de uso

Construct2 tiene distintos tipos de licencias:

- **Licencia gratuita:** esta edición permite probar el producto, imponiendo varias restricciones, principalmente en la exportación de proyectos, el límite de eventos (máximo 100) y el límite de capas (máximo 4).
- **Licencia personal:** permite utilizar todas las características del programa, aunque si el juego obtiene ganancias superiores a los U$S5000, exige pasar a una licencia empresarial.
- **Licencia empresarial:** orientado a empresas, no impone límites a la generación de ganancias.
- **Licencia educativa:** Este tipo de licencia permite su instalación en instituciones educativas, con precios variables según las características y el tiempo de utilización.

Conclusiones

Si bien por su facilidad de uso y la gran funcionalidad que ofrece, esta herramienta resulta ideal para la creación de prototipos y juegos simples, para la creación de juegos grandes presenta algunas dificultades. En este tipo de juegos, la programación conducida por eventos suele complejizarse. A medida que crece el número de objetos e interacciones entre ellos, también crece la cantidad de eventos. Cuando se tienen cientos de eventos, las colisiones entre los mismo se convierten en un problema y hacen difícil la tarea de programarlos. Aunque la posibilidad de incorporar código a través del SDK JavaScript puede ayudar a resolver ciertas cuestiones, esta funcionalidad está principalmente orientada a la programación de plugins y comportamientos, y no a la programación general de la lógica del juego.
Otra cuestión que puede surgir al programar juegos grandes con Construct2 es en relación con la eficiencia. Al estar oculto el código del motor del juego, ciertas optimizaciones de rendimiento inherentes al juego que se desarrolla no podrán ser llevadas a cabo.

4.4.2. Impact.js

Impact.js fue uno de los primeros motores de juego HTML5 que adquirió popularidad. Fue creado a fines del año 2010, en un momento en que la mayoría de los juegos web se desarrollaban en Flash y los juegos HTML5 recién empezaban a tomar algo de notoriedad.
A diferencia de lo que sucede con Construct2, para crear juegos a través de Impact.js es necesario escribir código. El motor consiste en un conjunto de archivos JavaScript que abstraen las distintas partes del motor. Para definir la lógica del juego, se deben programar los comportamientos de cada una de las entidades que participan, siguiendo una metodología orientada a objetos. Impact.js cuenta también con una herramienta de edición de niveles y facilidades para la exportación del juego.

Nivel de abstracción / Facilidad de uso

El conjunto de funciones que expone la librería es reducido. Esto no quiere decir que la funcionalidad que ofrece sea escasa e incompleta, por el contrario, este conjunto de funciones resulta versátil y suficiente para crear juegos de diversos tipos. Impact.js oculta completamente el manejo de recursos, los mecanismos de renderizado de gráficos, la entrada de usuario, y la organización de entidades y escenas. También ofrece un sistema de física y colisiones simplificado.
La documentación está redactada de manera simple y guiada, abarcando cada uno de los aspectos claves de la programación con esta librería. Incluye también videotutoriales e información de referencia de todas las clases y métodos disponibles.

Funcionalidad

Impact.js cuenta con las siguientes características:

- Facilidades para la creación de juegos de plataformas
- Manejo de animaciones
- Manejo de física básica
- Posibilidad de integración con motor de física avanzado Box2D
- Visualización de colisiones, velocidades y direcciones en modo depuración
- Parallax scrolling
- Fondos en mosaico
- Manejo de cámara
- Plugins creados por la comunidad

Herramientas para creación de contenido

Este motor de juego incluye un editor de niveles propio conocido como “Weltmeister Level Editor”. Esta herramienta está totalmente integrada con Impact.js y permite crear escenarios y niveles de manera sencilla a través de una interfaz web.
El editor permite diseñar fondos a través de mosaicos (Tiles), especificar qué elementos serán considerados como sólidos, y agregar y posicionar entidades ya

programadas. Además, permite conectar entidades entre sí para crear cadenas lógicas.

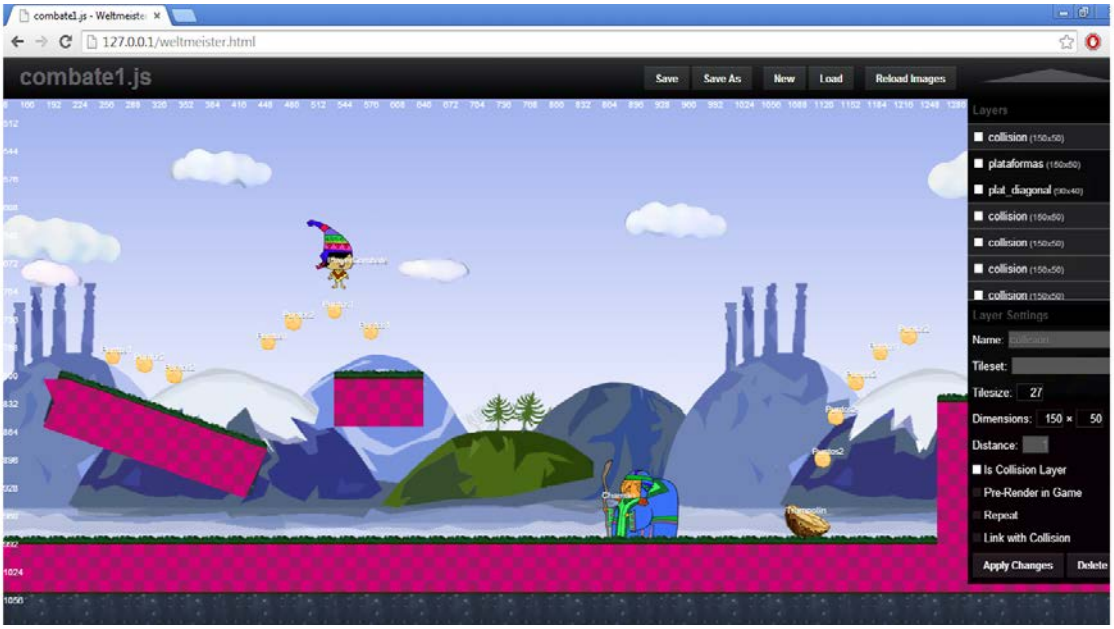


Fig 4.2: Editor de Niveles

Flexibilidad

Dado que se puede acceder al código fuente del motor, el grado de flexibilidad que se puede alcanzar es alto. La simplicidad con la que está escrito el motor y su separación en módulos facilita la tarea de agregar nueva funcionalidad o modificar características existentes. Aún el editor de niveles Weltmeister expone su código fuente y puede ser modificado.

Impact.js permite también la creación de plugins. Estos plugins, son porciones de código que permiten agregar funcionalidad al motor sin modificar de manera directa su código fuente. Además de permitir su creación, existen numerosos plugins ya desarrollados por la comunidad disponibles de manera gratuita. Entre las funcionalidades que ofrecen estos plugins se cuentan: algoritmos de detección de caminos, características multijugador, iluminación dinámica, implementación de comportamientos comunes, almacenamiento local y otros.

Portabilidad

Los juegos creados con Impact.js funcionan en cualquier navegador que soporte HTML5: Firefox, Chrome, Safari, Internet Explorer 9+ y Opera. Además, Impact.js está preparado para funcionar en Ejecta, un proyecto de código libre programado en Objective-C que permite compilar juegos Web para su ejecución de manera nativa en iOS.

El framework Ejecta toma el código JavaScript, lo ejecuta y renderiza el juego en pantalla utilizando OpenGL. También permite utilizar capacidades multitáctil y sonido a través de OpenAL.

Otro mecanismo de publicación en móviles, consiste en utilizar AppMobi's GameDev XDK, una plataforma de servicios en la nube integrada con Intel XDK, que permite empaquetar juegos creados con Impact.js de manera sencilla. Como resultado de este servicio, se obtienen aplicaciones nativas listas para ser publicadas en los mercados iOS y Android.

Licencia de uso

Impact.js se ofrece a través de una licencia comercial que incluye:

- El motor de juego
- El editor de niveles Weltmeister
- El código fuente completo del motor y del editor de niveles.
- El framework Ejecta
- Juegos de demostración con código fuente (Drop, Simple Jump'n'Run, Box2D Physics, Pong)

El pago de la licencia, permite la libre comercialización y distribución de los juegos creados con la herramienta, pero no así del código fuente original del motor.

Conclusiones

Impact.js es un motor simple de usar con el que se pueden desarrollar gran variedad de juegos, aunque presenta algunas limitaciones como por ejemplo el hecho de no contar con un cargador de recursos dinámico, o con un manejo de cámara complejo que permita realizar zooms. Otra característica no incluida es la opción de renderizar gráficos a través de WebGL, que en ciertas configuraciones

supone una mejora importante en cuanto a rendimiento. Estas funcionalidades, aunque interesantes, no serán imprescindibles para la mayoría de los juegos que se desarrollen, y en el caso de ser necesarias, dado que el código del motor es accesible, podrían ser implementadas.
A diferencia de Construct2 que utiliza un sistema de programación orientado a eventos, el mecanismo de programación que utiliza Impact.js sigue un enfoque más tradicional, similar al del resto de los motores presentes en el mercado, por lo que puede servir como punto de partida en el aprendizaje de la programación de videojuegos, antes de lanzarse a opciones más complejas.

4.4.3. Playcraft

La versión beta de este motor de juego fue lanzada en el año 2012, apuntando a simplificar la creación de juegos HTML5 de nivel profesional. Para lograr esto, el motor propone la utilización de una metodología de programación conocido como “Entidades-Sistemas” que busca solucionar algunos problemas que surgen al programar juegos complejos utilizando un modelo tradicional basado en clases y herencia.
Si bien el motor continúa en estado beta, muchas de sus características ya están implementadas y funcionan correctamente.

Nivel de abstracción / Facilidad de uso

Para la gente habituada a la programación orientada a objetos, adaptarse al modelo “Entidades-sistemas” puede requerir un esfuerzo adicional. Dado que todos los ejemplos que presenta el motor de juego están estructurados según este paradigma, adquirir un conocimiento claro de su funcionamiento requerirá más tiempo que otras opciones.
Para aliviar esta cuestión, la documentación es completa. Se ofrece por un lado un tutorial guiado con diversos ejemplos, y por otro la especificación completa de clases y métodos del motor.
Aunque con un grado de complejidad mayor que Impact.js, este motor también abstrae la administración de recursos, los mecanismos de renderizado, la creación y manipulación de entidades, además de cuestiones específicas de

ciertos tipos de juego, como el hecho de poder identificar si un objeto se encuentra parado sobre un sólido.
Si bien la incorporación de un motor de física complejo agrega versatilidad, también incorpora un nivel de dificultad al programar los movimientos de las entidades.

Funcionalidad

Este motor actualmente cuenta con las siguientes funcionalidades ya implementadas:

- Manejo de animaciones
- Manejo de física compleja
- Posibilidad de crear mapas de colisiones basados en polígonos
- Parallax scrolling
- Fondos en mosaico
- Sistema de partículas
- Manejo de cámara

Aunque no están disponibles todavía, un conjunto mayor de funcionalidades se encuentra anunciado:

- Multijugador
- Implementación de elementos de interfaz de usuario
- Integración con Facebook
- Editor de niveles propio

Herramientas para creación de contenido

Actualmente, este motor no cuenta con una herramienta propia para crear escenas y niveles. En su lugar, ofrece la posibilidad de importar mapas creados a través de Tiled Map Editor, una herramienta de código libre para la creación de escenarios. La integración con esta herramienta es, sin embargo, limitada: al

momento de cargar cada nivel es necesario referenciar explícitamente las entidades que aparecen en el mismo.

Flexibilidad

Al igual que como ocurre con Impact.js, al estar accesible el código fuente del motor, pueden realizarse todas las modificaciones que sean necesarias. Los distintos módulos del motor se encuentran correctamente comentados y poco acoplados unos con otros, lo que facilita su modificación. A diferencia de Impact.js y Construct2, este motor no permite la incorporación de plugins.

Portabilidad

Aunque el motor está preparado para abstraer su funcionamiento en distintos dispositivos y resoluciones de pantalla, por el momento este soporte se limita únicamente a navegadores. Si bien esto permitiría su ejecución en navegadores de celulares y tabletas, el rendimiento que ofrecen los mismos al ejecutar juegos está lejos del que se obtiene a través de aplicaciones nativas.
Para remediar esto, se ofrece cierto tipo de compatibilidad con CocoonJS, pero la misma está desactualizada y en muchos casos resulta deficiente.

Licencia de uso

La licencia del motor se ofrece de manera gratuita mientras el producto se encuentre en estado beta, aunque no se especifica si en algún momento se cobrará por el uso del motor. En cualquier caso, no se permite incorporar el código fuente del motor en los juegos desarrollados, aunque sí una versión ofuscada del mismo. Por el momento, el código fuente está disponible en GitHub y permite los aportes de la comunidad para su desarrollo.

Conclusiones

Si bien las funcionalidades que presenta este motor resultan interesantes para la creación de juegos complejos, no se han publicado nuevas versiones del mismo

desde Febrero de 2013 y no existe información concreta sobre la continuidad del proyecto. Al estar en estado beta, el código aún presenta errores, obligando al desarrollador a buscar soluciones a los mismos o a evitar utilizar características defectuosas. Por otro lado, el renderizado de gráficos no soporta WebGL, ni se encuentra lo suficientemente optimizado para funcionar en computadoras de gama baja. Estas cuestiones, sumadas a la complejidad inherente del motor, hacen que esta opción, al menos por el momento, no resulte atractiva para los desarrolladores que se inician en la programación de videojuegos. Aún para los desarrolladores experimentados, el hecho de que continúe en beta y no haya recibido actualizaciones en tanto tiempo, supone una razón suficiente para desalentar su uso.

4.4.4. Turbulenz Engine

Este motor de juego 2D/3D fue lanzado públicamente en 2012, aunque su popularidad creció a partir del anuncio de la apertura de su código fuente en Abril de 2013.
Según los creadores de este motor, “Los principales objetivos detrás del diseño de Turbulenz Engine son rendimiento, modularidad y personalización. Los usuarios del motor deberían ser capaces de construir cualquier tipo de juego sin limitaciones, de manera eficiente y logrando un producto final con un rendimiento óptimo tanto durante la carga, como durante el juego.”
Existen principalmente dos maneras de trabajar con el motor Turbulenz, por un lado, referenciando directamente los archivos JavaScript de los distintos módulos, a modo de librería; y por otro, utilizando un SDK de desarrollo que incluye un servidor propio, herramientas de empaquetado, documentación y ejemplos.

Nivel de abstracción / Facilidad de uso

El grado de abstracción que ofrece este motor es bajo. Al menos para la creación de juegos 2D, el motor no abstrae el manejo de escenas, el ordenamiento de entidades, el manejo de animaciones, la estructura básica de llamado a los distintos módulos, ni la implementación de comportamientos comunes a juegos. Todas estas cuestiones quedan en manos del desarrollador y deben ser

diseñadas y programadas antes de desarrollar el código correspondiente a la lógica del juego. Esta tarea conlleva tiempo y agrega complejidad al desarrollo, aunque permite un nivel de personalización muy alto.

Funcionalidad

Gran parte de la funcionalidad que ofrece este motor está centrada en la creación de juegos 3D, sin embargo, también se incluyen características orientadas a juegos 2D:

- Manejo de física compleja
- Carga de recursos bajo demanda
- Posibilidad de crear mapas de colisiones basados en polígonos
- Visualización de colisiones, velocidades y direcciones en modo depuración
- Manejo de cámara
- Renderizado eficiente (WebGL)
- Efectos gráficos WebGL (distorsión, color, bloom y desenfoque gaussiano)

Además del motor de juego, Turbulenz ofrece una plataforma de desarrollo, testeo y publicación de juegos. En caso de que el juego esté orientado a funcionar bajo esta plataforma, se ofrece la siguiente funcionalidad:

- Multijugador
- Simulador de latencia de red
- Salvado de progreso del usuario
- Manejo de tablero de puntuaciones
- Manejo de insignias

Herramientas para creación de contenido

Si bien existe un editor[12] de escenarios diseñado para crear juegos 3D con Turbulenz, el mismo no resulta adecuado para la creación de juegos 2D. Para este tipo de juegos, no se ofrecen herramientas especiales de creación de

[12] https://github.com/turbulenz/gargantia_editor

contenido. A pesar de esto, Turbulenz incluye herramientas que simplifican el empaquetado del juego y el manejo de assets, incluyendo utilidades para la compactación de librerías JavaScript, la compresión de imágenes PNG y la generación automática de mipmaps, entre otras.

Flexibilidad

Las funcionalidades que implementa Turbulenz son de bajo nivel y orientadas específicamente a ocultar aspectos técnicos generales, por lo que el desarrollador alcanza un alto grado de libertad, pudiendo organizar el trabajo con el motor según sus preferencias. El desarrollador deberá elegir sobre qué paradigma se construirá el juego y de qué manera se conectarán los distintos módulos.
Debido a que las tareas que realiza el motor son técnicas y generales, comúnmente no será necesario realizar modificaciones en el código del motor para agregar nueva funcionalidad. Probablemente esta funcionalidad deba ser construida por encima de las funciones que ya ofrece el motor. De igual manera, en caso de ser necesario, el código del motor se encuentra abierto y puede ser modificado.

Portabilidad

Para el renderizado de gráficos, Turbulenz hace uso de la tecnología WebGL, por lo que la compatibilidad con navegadores de manera nativa se encuentra disponible solo para Google Chrome, Firefox 10+, Safari 5.1+ e Internet Explorer 11.
En el caso de los navegadores que no soportan WebGL, Turbulenz posibilita su ejecución mediante la instalación de un plugin adicional. En estos casos, el plugin inyecta todas las APIs faltantes directamente dentro del motor JavaScript nativo, posibilitando que el juego cuente con todas las funcionalidades requeridas. Esta extensión se provee exclusivamente por cuestiones de compatibilidad.
Con respecto a la portabilidad en dispositivos móviles, Turbulenz ofrece una herramienta para ejecutar los juegos creados con el motor, de manera nativa en Android. Una herramienta similar para permitir el soporte nativo en iOS ha sido aununciada, aunque la misma todavía no está disponible

Licencia de uso

Todo el código del motor de Turbulenz se encuentra publicado bajo licencia MIT. Esta licencia es libre y no impone restricciones. Permite entre otras cosas modificar el motor, combinarlo con otros frameworks, distribuirlo, venderlo o publicarlo libremente. Los juegos creados con este motor no tienen tampoco restricciones comerciales de ningún tipo y pueden ser publicados bajo cualquier plataforma.

Conclusiones

Turbulenz Engine está orientado a crear juegos sin limitaciones y ofrece toda la funcionalidad que un juego complejo puede requerir. En cuanto a la creación de juegos sencillos, es importante notar que el motor carece de abstracciones de alto nivel, por lo que estos desarrollos demandarán más tiempo y requerirán un conocimiento general sobre arquitectura de juegos. En estos casos, otras alternativas pueden resultar más adecuadas.
Una cuestión a tener en cuenta si se pretende crear un juego que sea compatible con distintos rangos de dispositivos, es la de la utilización de WebGL. Si bien el uso de esta tecnología en general produce un notable incremento en el rendimiento (en comparación con el uso del elemento Canvas), en ciertos dispositivos de gama baja, o no está disponible, o dispara el consumo de recursos resultando en un rendimiento pobre. La utilización del plugin ofrecido por Turbulenz sirve para remediar en parte este problema, pero impone el costo de tener que instalar software adicional en el navegador.

4.4.5. Conclusiones generales

A lo largo de este capítulo se han analizado distintos motores de juego HTML5 con características diversas. De este análisis se desprende que Construct2 e Impact.js resultan los más adecuados para la creación de prototipos y juegos simples, aunque ambos presentan licencias pagas. El uso de Turbulenz para la

creación de este tipo de juegos no parece ser la mejor opción debido a su complejidad y bajo nivel de abstracción. Sin embargo, su potencial para crear juegos complejos y las características liberales de su licencia pueden hacerlo atractivo para distintos desarrollos. El motor Playcraft, por otra parte, aunque promisorio debido a la funcionalidad anunciada, por el momento se encuentra incompleto y no ofrece la robustez de otras alternativas, ampliamente testeadas y con mayor cantidad de usuarios.

A modo de síntesis, en la Figura 4.3 se presenta una tabla en la que se expresa el grado en que cada motor satisface los criterios analizados. El grado de funcionalidad no ha sido incluido debido a que el mismo es subjetivo y dependerá en gran medida de las necesidades de cada proyecto.

	Construct2	Impact JS	Playcraft Engine	Turbulenz Engine
Abstracción	Muy alto	Alto	Intermedio	Bajo
Flexibilidad	Intermedio	Alto	Alto	Muy alto
Integración con herramientas	Muy alto	Alto	Intermedio	Bajo
Portabilidad	Muy alto	Alto	Intermedio	Intermedio
Licencia	Comercial	Comercial	Libre (con restricciones)	Libre

Fig 4.3: Cuadro comparativo de motores evaluados

Aunque aquí se han analizado estos cuatro motores, existen también muchas otras alternativas. El desarrollo de juegos con HTML5 es relativamente nuevo y actualmente se encuentra en el centro de la atención, por lo que nuevos motores surgen continuamente.

A pesar de no haber sido probados para la implementación de Raíces, haremos una breve reseña de dos alternativas lanzadas recientemente y que pueden servir para desarrollos futuros: Phaser y Unity3D (a partir de la versión 4.3 con soporte 2D).

En el caso de Phaser, este motor se lanzó a principios de 2014 y adquirió gran popularidad en pocos meses. Sus principales fortalezas son su simplicidad de uso y su alta portabilidad a dispositivos móviles. La funcionalidad que ofrece también

es amplia y se encuentra en el orden de la ofrecida por Impact JS. El código del motor se encuentra abierto y se ofrece bajo licencia MIT.

Unity 3D es un motor de videojuegos y plataforma de desarrollo de amplia trayectoria y popularidad, orientado a la creación de juegos en tres dimensiones. A partir de la versión 4.3, lanzada a fines de 2013, incluye soporte nativo y herramientas integradas para la creación de juegos 2D. Aunque Unity se ofrece bajo una licencia comercial, es posible descargar también una versión limitada gratuita. El grado de funcionalidad y la diversidad de herramientas que incluye este motor, posibilita la creación de juegos 2D/3D profesionales, sin limitaciones. En su última versión, Unity 5 incluye un mecanismo de exportación a WebGL, característica que lo posicionaría como en una alternativa a Turbulenz para el desarrollo de juegos HTML5 complejos.

5

5. Implementación

5.1. El proceso

La implementación de Raíces ha seguido un enfoque iterativo. En una primera etapa, se crearon prototipos simples para empezar a definir características de la mecánica elegida y testear su funcionamiento. La creación de estos prototipos, supuso una implementación básica de un nivel de cada subgénero: Combate, Correr y Pensar. Para esto se utilizó el motor Construct2 y gráficos provisionales.
Estos primeros prototipos fueron puestos a prueba desde un primer momento, permitiendo así identificar de manera temprana las fortalezas y debilidades del juego propuesto. La flexibilidad y facilidad de uso de Construct2 hacen de este software una herramienta ideal para la creación de prototipos, permitiendo el mismo realizar ajustes rápidos en el diseño de niveles y variables de la mecánica, logrando una mejora progresiva del nivel de jugabilidad (es decir, de las sensaciones positivas que experimenta el jugador al interactuar con el sistema de juego).
Esta dinámica de trabajo se ajusta perfectamente al diseño de juegos utilizando el modelo MDA. A partir de la identificación de las emociones y dinámicas generadas al momento de ser jugados los prototipos, ciertas mecánicas específicas se corrigen o descartan, poniéndose a prueba otras nuevas en la siguiente versión.
El proceso de prototipado utilizando Construct2 se extendió durante varias semanas hasta lograr una versión que mostraba una buena aceptación por parte de los jugadores. A pesar de las facilidades y funcionalidades que ofrecía este software para el desarrollo rápido de juegos, en vistas de una creciente complejidad del diseño del juego propuesto, empezaron a plantearse ciertos problemas. El mecanismo de programación orientada a eventos que utiliza

Construct2, al escalar la complejidad del juego (cantidad de niveles, variedad de mecánicas, integración in-game con redes sociales, interfaz de usuario compleja, etc.) no resultaría el más adecuado para la realización integral de Raíces.

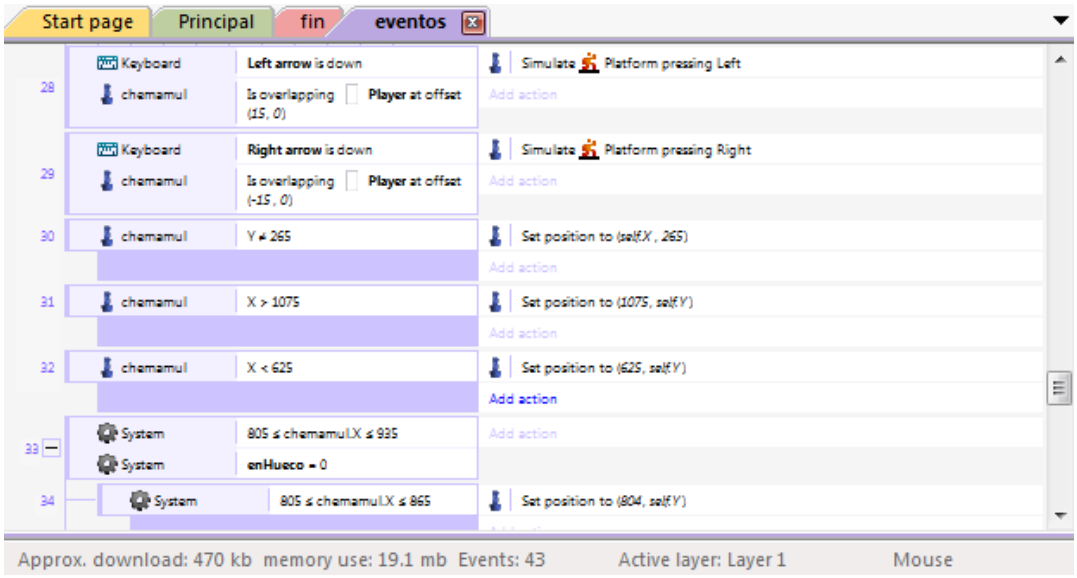


Fig. 5.1: Sistema de eventos de Construct2

Partiendo de este prototipo básico de un nivel por subgénero, se empezó a realizar una implementación algo más desarrollada del juego (con varios niveles por subgénero) utilizando otro motor: Impact.js.

La transición a este motor, requirió una re-implementación completa de las mecánicas y niveles existentes. La forma de programar juegos con Impact.js, al seguir un paradigma orientado a objetos, difiere enormemente de la forma de trabajo con Construct2. Más allá de lo relativo a la implementación, las cuestiones de diseño de juego desarrolladas durante el trabajo con Construct2 constituyeron un gran aporte y sirvieron de base para el desarrollo inicial con Impact.js.

La posibilidad de escribir código, motivó la generación de comportamientos más complejos y reutilizables. En esta etapa, el prototipo fue adquiriendo una forma más cercana a lo que sería el juego final: varios niveles encadenados, de dificultad creciente, y con entidades claramente definidas y funcionales a las necesidades de cada nivel y del juego como un todo. En esta etapa, también comenzó a implementarse una interfaz de usuario básica.

Fig. 5.2: Implementación del prototipo con Impact.js

Si bien el avance utilizando este motor fue grande, las restricciones comerciales de su licencia traían aparejada la imposibilidad de distribuir el código fuente del motor. Debido a las características académicas de este desarrollo, el uso de este motor para la implementación definitiva del juego no era la mejor opción. A pesar de las dificultades que supone el hecho de implementar nuevamente un juego utilizando otro motor, esta tarea permite adquirir un conocimiento más profundo de las características de los motores y del desarrollo de juegos en general. En base a esta experimentación con distintos motores, se ha desarrollado el capítulo comparativo presentado anteriormente en este trabajo.

En la búsqueda de un nuevo motor para realizar una re-implementación de Raíces, una de las opciones más atractivas en ese momento era la del motor de juego Playcraft. Aunque aún se encontraba en desarrollo, presentaba una licencia libre y gran potencial. En vista de esto, empezaron a realizarse pruebas, implementando un nivel con algunas mecánicas básicas.

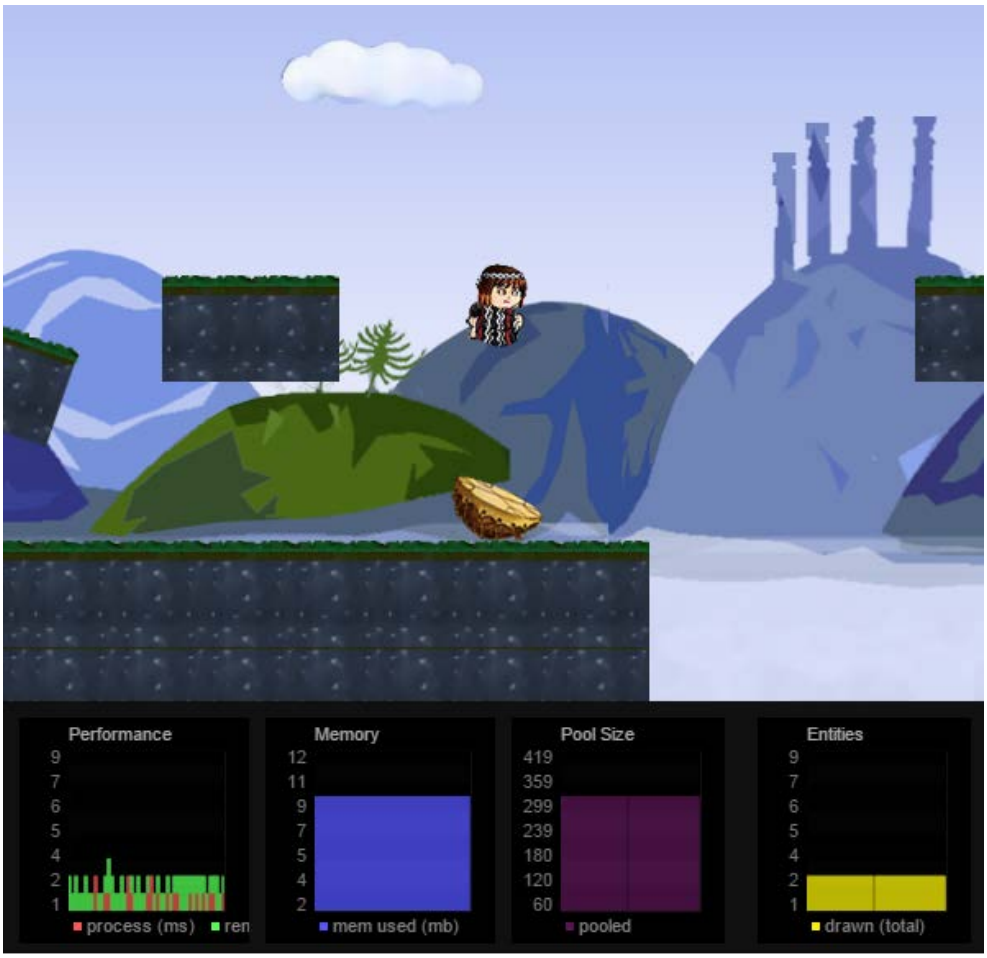


Fig. 5.3: Implementación básica realizada con Playcraft

Playcraft poseía características interesantes, sin embargo la gran presencia de errores y el pobre rendimiento gráfico al ejecutar el juego en ciertas computadoras, motivaron la búsqueda de una alternativa más robusta y eficiente.

Después de evaluar las características de otras alternativas de código libre disponibles, se decidió probar el motor de juego de código abierto Turbulenz. Aunque relativamente nuevo, contaba con algunos juegos de gran calidad como carta de presentación y ejemplos que demostraban su alto potencial. Las primeras pruebas realizadas con este motor arrojaron buenos resultados en lo referente a rendimiento, funcionalidad y robustez, por lo que se definió su utilización.

En particular, Turbulenz mostró una de sus principales fortalezas en su motor de física, que aun ofreciendo un amplio rango de posibilidades, resulta eficiente y liviano. Otros módulos de bajo nivel con los que se obtuvieron resultados satisfactorios fueron los módulos de manejo de recursos y corrección de errores.

En contraposición a esto, el módulo de renderizado no siempre se mostraba del todo eficiente. Al avanzar una re-implementación de un nivel de Raíces utilizando este motor, se observó que el proceso de renderizado demoraba más tiempo que algunas de las otras alternativas probadas previamente. Pensando a futuro, con una complejidad gráfica creciente y una necesidad de compatibilidad con distintas computadoras, esta desventaja se iría profundizando. En adición a esto, Turbulenz no resulta compatible de manera nativa con navegadores sin WebGL, restringiendo aún más la compatibilidad. Esta dependencia de WebGL se producía únicamente debido al motor de renderizado.

Frente a estas cuestiones, se empezó a gestar la idea de reemplazar este aspecto conflictivo de Turbulenz utilizando un motor de renderizado más eficiente e independiente de WebGL. La organización de Turbulenz en módulos poco acoplados hacía viable este reemplazo.

Entre los motores de renderizado HTML5 disponibles en ese momento, Pixi.js[13] destacaba sobre el resto por ciertas características: de código abierto, orientado a la eficiencia y multiplataforma. Esta librería de renderizado, ofrece además un conjunto variado de funcionalidades:

- Detector automático de renderizado (utilizando WebGL en caso de que esté disponible y Canvas en caso contrario, asegurando compatibilidad con todos los navegadores con soporte HTML5)
- Grafo de escena completo (permite organizar los elementos a renderizar en distintos niveles jerárquicos)
- Soporte para atlas de texturas
- Cargador de assets y tiras de imágenes
- Manejo de interacción con Mouse y pantallas multi-táctiles
- Renderizado de texto (nativo y BitmapFonts, multilínea)
- Renderizado de texturas
- Integración con Spine (para creación de animaciones esqueléticas)
- Dibujo de primitivas
- Manejo de máscaras
- Aplicación de filtros (Displacement, Blur, Pixelate, Invert, Sepia, Twist, etc)

[13] http://www.pixijs.com/

Pixi.js, en comparación con el motor de renderizado de Turbulenz, ofrecía numerosas ventajas, por lo que se decidió su utilización.
Para realizar el reemplazo del módulo de renderizado, era necesaria la creación de una arquitectura que gestionara la integración de Pixi.js y el resto de los módulos de Turbulenz de manera transparente. Esta arquitectura, debía además, proveer soporte para la creación de entidades de juego y niveles de una manera sencilla.
Para el desarrollo final de Raíces, se decidió, por lo tanto, crear un motor de juego propio, escrito en JavaScript y basado en Pixi.js y Turbulenz, el cual brindara abstracciones de alto nivel y supliera las carencias de este último en relación a los aspectos de renderizado, compatibilidad con distintos navegadores y equipos, portabilidad, e integración con herramientas externas (i.e. editor de niveles, editor de diálogos, generadores de contenido multimedia).

5.2. Arquitectura del modelo de entidades de juego

Al momento de ejecutarse, un juego estará formado por un conjunto de objetos que interactúan entre sí para dar vida a las mecánicas establecidas. Para referirnos a estos objetos en particular, que hacen a la implementación específica de cada juego y funcionan por encima del motor, utilizaremos el término entidades de juego, o simplemente entidades. La forma de programar estas entidades, aunque en algunos casos existan ciertas libertades, estará en gran medida determinada por la arquitectura que el motor de juego imponga.
Cuando se trata de diseñar una arquitectura que soporte la creación de entidades de juego siguiendo un enfoque tradicional orientado a objetos, en un primer momento se piensa en la creación de una clase padre "Entidad", a partir de la cual se derivarán las diversas subclases que representan cada elemento lógico del juego. Estas clases se irán organizando en sucesivas subclases dando lugar a una taxonomía que represente todo el modelo de entidades.
Si bien este enfoque funciona correctamente para la creación de la mayoría de los sistemas tradicionales, en el caso del desarrollo de juegos, los constantes cambios en los requisitos y funcionalidad, hacen que esta estrategia no sea la

más adecuada. J. Gregory [34] identifica algunos problemas que surgen al utilizar jerarquías anchas y profundas para el modelado de entidades de juego:

- **Dificultad en el entendimiento, mantenimiento y modificación de las clases:** cuanto más profunda es una clase, más difícil será de entender. Para entender una clase, habrá que entender también todas sus clases padre.
- **Incapacidad para describir taxonomías múltiples:** el mecanismo de herencia simple solo permite agrupar las clases según un único criterio. Si bien el uso de herencia múltiple podría solucionar este problema, las complicaciones que trae aparejado su uso, al menos en lo que respecta al desarrollo de videojuegos, por lo general superarán a sus beneficios.
- **Efecto de burbujeo hacia arriba:** cuando se inicia el diseño de la jerarquía, las clases raíces son simples y exponen poca funcionalidad. A medida que se agrega más funcionalidad al juego, la necesidad de compartir código entre dos o más clases que están en diferentes subárboles, hace que esta funcionalidad se mueva a clases superiores de la herencia.

Una solución a estos problemas, consiste en el aislamiento de las distintas funcionalidades en clases independientes, denominadas **componentes**, que proveen un servicio único y bien definido. Estos componentes funcionarán en relación a **sistemas**, que se encargan de realizar cierto procesamiento sobre estos componentes. Cada clase, en lugar de estar definida por su ubicación en el árbol de jerarquía y el mecanismo de herencia, estará definida por los componentes que tenga asociados, utilizando un mecanismo de composición. Esta arquitectura de programación de entidades de juego se conoce como "**Entidades-Sistemas**".

Llevando este paradigma al extremo, todos los comportamientos podrían ser movidos a componentes evitando completamente la subclasificación de Entidades. En este enfoque completamente puro del modelo "Entidades-Sistemas", la creación de las entidades se delega en una clase Factory (utilizando el patrón del mismo nombre) que tenga funciones para la construcción de los distintos tipos de entidades, asignándole a cada una los componentes que les

correspondan. También será necesario definir sistemas para manejar el comportamiento de cada entidad. Si bien esta forma de trabajo es utilizada por algunos motores de juego, la misma resulta poco intuitiva para aquellos habituados a la programación orientada a objetos. Un ejemplo de motor que utiliza un enfoque "Entidades-Sistemas" puro, es el caso de Playcraft Engine, analizado en el capítulo anterior.

La arquitectura adoptada por el motor que da soporte a Raíces, en cambio, si bien posee características del patrón Entidades-Sistemas, no llega al extremo de tener todos los objetos creados como instancias de una única clase. En su lugar, presenta un esquema mixto: clases concretas con comportamiento propio, que pueden tener además componentes asociados. En esta arquitectura, existirán tantas clases como tipos de objetos existan en el juego. Cada una de estas clases, podrá definir su comportamiento desde la propia definición de la clase.

Al desarrollar los objetos que participan del juego, en la mayoría de los casos se crearán clases que heredan de "Entity", una clase que representa una entidad genérica con los componentes Body, Sprite y Animation. Estos componentes, serán administrados de manera automática, por tres sistemas: sistema de manejo de física y colisiones, sistema de renderizado de sprites y sistema de animaciones, respectivamente. En cada iteración del juego, se invocarán estos sistemas procesando los componentes de todas las entidades activas y actualizando su estado. Al crear entidades, también existe la posibilidad de extender la clase “EmptyEntity”, que representa una entidad vacía, a la que luego se le pueden asociar los componentes específicos que se requieran.

Esta arquitectura, que tiene algunos puntos en común con la estructura básica del modelado de entidades utilizado por la gran mayoría de los motores de juego tradicionales, y por Impact.js en particular, mantiene un enfoque orientado a objetos y permite una dinámica de trabajo eficiente.

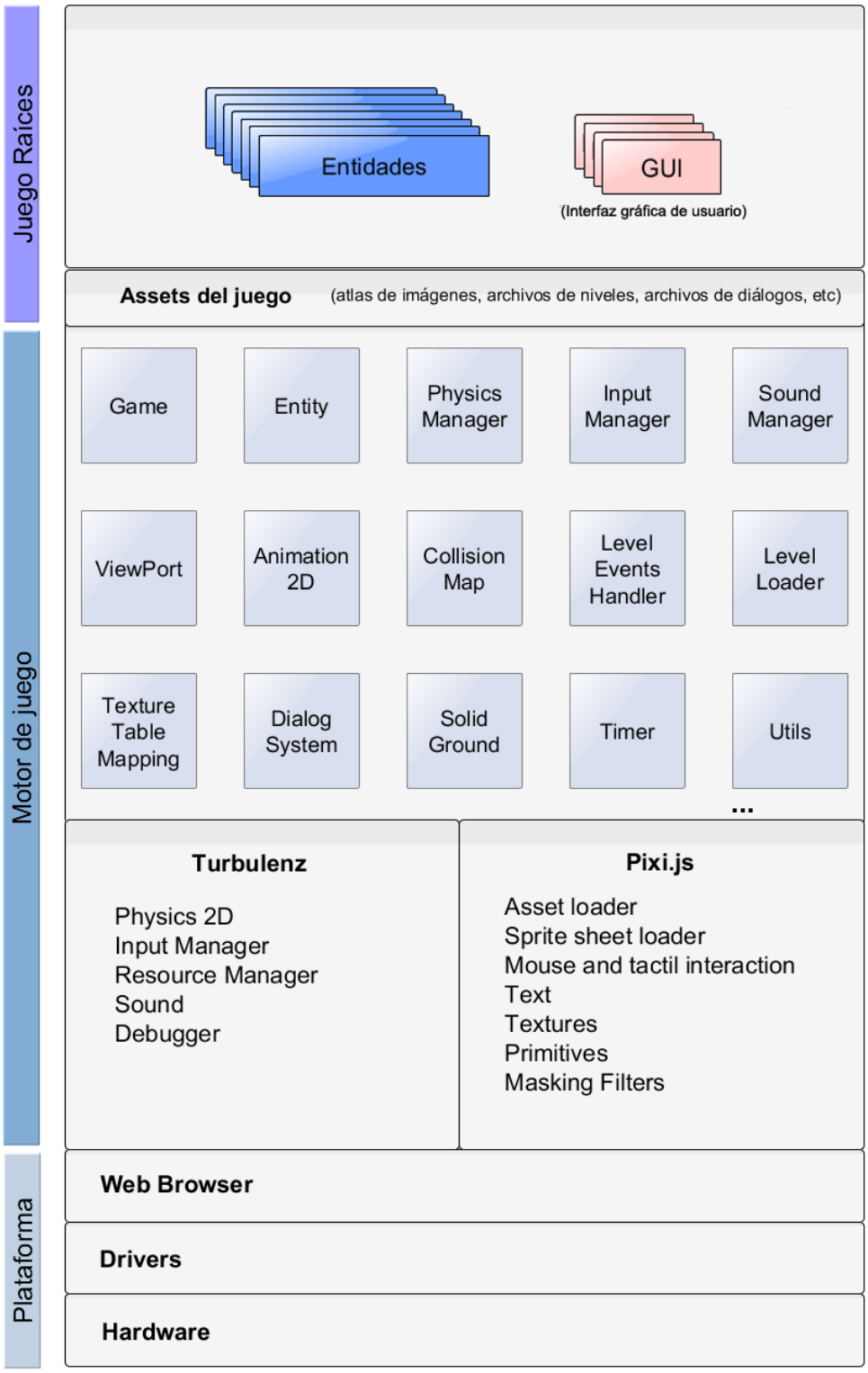


Fig. 5.4: Diagrama del motor de juego que da soporte a Raíces

5.3. Módulos de alto nivel

Para facilitar la implementación del videojuego de plataformas Raíces, fue necesaria la creación de diversos módulos con abstracciones de alto nivel. Estos módulos, se encargan de ocultar aspectos técnicos y encapsulan ciertas funcionalidades para simplificar la creación de los distintas partes que componen el juego. Si bien estos módulos han sido creados especialmente para sustentar el desarrollo de Raíces, los mismos podrían a su vez ser reutilizados en otros desarrollos de juegos y no se limitan únicamente al género de plataformas.
En las secciones siguientes se hará una descripción de la funcionalidad y características esenciales de cada uno de los módulos que componen el motor de juego desarrollado.

5.3.1. Entity Manager

La clase Entity constituye la estructura base de todos los objetos que participan del juego y tienen una imagen, una animación y un cuerpo físico asociados. Cada entidad tendrá también, en la mayoría de los casos, un comportamiento asociado.
Este módulo brinda las bases para permitir la creación de entidades de una manera sencilla, y abstrae comportamientos comunes a todas las entidades de juego:

- Gestión de la relación entre el cuerpo físico de la entidad, correspondiente al motor de física de Turbulenz, y la clase Sprite, correspondiente a Pixi.js y encargada almacenar su representación gráfica. Esta gestión automática y transparente al programador de entidades, posibilita que cada entidad se dibuje correctamente en pantalla de acuerdo a la posición y rotación determinada por el cuerpo físico.

- Gestión de mipmaps. Como describe S. Rabin [35], la técnica conocida como mipmapping se basa en utilizar, en lugar de una única textura, una secuencia de texturas, cada una de ellas de aproximadamente la mitad del

tamaño de la anterior, con el fin de reducir los efectos del aliasing[14]. Este conjunto de texturas se conoce como cadena de mipmaps.

En el caso de los juegos 2D, los efectos del aliasing ocurren cuando se intenta redimensionar una imagen de alta resolución en una imagen de resolución mucho menor, generando artefactos no deseados. Para evitar esto, las imágenes de las entidades estarán disponibles en cuatro resoluciones distintas (*small*, *normal*, *big*, *bigger*). La gestión de mipmaps consiste en elegir, de acuerdo a la resolución y zoom de juego actual, la textura base más adecuada para dibujar la entidad.

– Actualización de los límites que determinan el fragmento de imagen a dibujar, en función del cuadro de animación actual.

– Control de variables asociadas a la representación gráfica de la entidad:
  - Offset (distancia en X e Y del cuerpo físico respecto a la imagen)
  - Flipped (determina si la entidad debe dibujarse espejada o no)
  - Resize (permite aumentar o disminuir el tamaño de la imagen)
  - zIndex (determina el orden de profundidad con el que debe dibujarse la entidad)

– Preparación de la entidad para gestionar colisiones.

– Manejo de aspectos de optimización.

– Definición de métodos auxiliares para:
  - Posicionar una entidad leída desde un archivo de nivel
  - Aplicar filtros de color, brillo y saturación a la representación visual de la entidad
  - Determinar si una entidad se está mostrando actualmente en pantalla o no

Dado que casi todos los objetos del juego serán construidos como subclases de Entity, el grado de funcionalidad abstraída por este módulo, determina en gran parte la simplicidad con la que se definirán posteriormente estos objetos para implementar un juego concreto.

[14] El *aliasing* es el defecto gráfico que hace que, en una pantalla o imagen, ciertas curvas y líneas inclinadas presenten un efecto visual tipo escalón.

5.3.2. Game Manager

El administrador de juego cumple la importante tarea de asegurar la ejecución del videojuego como un todo. Este administrador funciona como el controlador principal de la ejecución del juego y se comunica con el resto de las partes del motor.

Las funciones que desempeña el administrador de juego son muchas y variadas:

- Definición de parámetros de configuración global (versión a ejecutar, tamaño de pantalla inicial, FPS -fotogramas por segundo-, modo desarrollo/producción)
- Definición de nivel inicial y lista de niveles (en el caso de Raíces agrupados por subgénero)
- Carga inicial de recursos (i.e. imágenes, sonidos, niveles) y generación de feedback visual sobre el avance de esta carga (barra de progreso)
- Creación e inicialización de subsistemas (renderizado, física, sonido, entrada de usuario, mapa de colisiones, etc.)
- Inicio de la ejecución del juego una vez completada la carga de recursos
- Control del loop del juego:
    1. Actualización de entidades
    2. Actualización de mundo físico
    3. Renderizado gráfico general
    4. Lógica específica al detectarse finalización de niveles u otras condiciones

A partir del administrador de juego (inicializado en la variable *raices.game*), se permite un acceso global a todos los subsistemas, entidades y parámetros de configuración que hacen a la ejecución de Raíces.

Dado que solo habrá un único administrador de juego en un determinado momento, existirá a su vez una única instancia de cada subsistema. Estas dos características (acceso global e instancia única) hacen que los subsistemas funcionen como implementaciones del patrón Singleton [36].

Si bien no se establecen limitaciones en el acceso directo a los miembros de la clase Game, para la modificación de algunos de ellos se recomienda el uso de métodos. Un ejemplo de esto, es el caso de la lista de entidades en ejecución: si bien se permite su modificación directa, para asegurar una correcta ejecución del juego, en el momento de querer agregar o eliminar una entidad, deben utilizarse los métodos auxiliares correspondientes.

5.3.3. Módulos específicos

- **Physics Manager**

  Desde este módulo pueden definirse los límites que tendrá el escenario de juego y la frecuencia de actualización de las entidades físicas. Encapsula el motor de física provisto por Turbulenz.

- **Input Manager**

  Encapsula los mecanismos de entrada provistos por Turbulenz para el manejo de interacción con teclado y mouse, y los provistos por Pixi.js para el manejo de interacción táctil.

  Es el encargado de guardar para cada iteración del juego, las teclas y botones presionados para su posterior consulta por las distintas entidades de juego. También gestiona acciones globales asociadas a teclas especiales.

- **Sound Manager**

  Ofrece una interfaz simplificada para la creación y reproducción de sonidos por parte de las entidades de juego, y para la reproducción de música global. Se encarga de controlar la carga de recursos de audio, la modificación de volúmenes y la aplicación de efectos de sonido.

- **ViewPort**

  Este módulo es el encargado de guardar el estado actual de la cámara del juego (zoom y posición) y del tamaño de la pantalla. Es utilizado internamente por el motor al momento de determinar los tamaños y posiciones de renderizado de las entidades de juego.

Expone funcionalidad para simular un efecto de temblor en la cámara, redimensionar el tamaño de la pantalla, y modificar el zoom durante el juego (de manera directa, o progresivamente a través de funciones easing).

- **Animation2D**

  La gran mayoría de las entidades de juego, tendrán asociadas al menos una animación formada por uno o más cuadros (denominados fotogramas). El módulo de animaciones 2D provee abstracciones para la creación de cada una de estas animaciones y sus variables de control asociadas (i.e. velocidad, repetición).

  Encapsula también funcionalidad que permite pasar una animación a otra, y avanzar una animación en función del tiempo transcurrido, de manera transparente para el programador de entidades**.**

- **Collision Map**

  El mapa de colisiones define la parte sólida de cada nivel, es decir, los espacios en los cuales el jugador podrá o no desplazarse. Almacena tamaño, forma y ubicación de todas las figuras de colisión que se definen desde el editor de niveles y constituyen el mapa de colisiones. Posee métodos auxiliares tanto para la creación como para la identificación de las zonas de colisión.

  Se encarga también de especificar los distintos grupos en los que se organizarán las entidades de juego que poseen un componente físico y pueden potencialmente colisionar con otras entidades.

- **Level Loader**

  La definición de los niveles o escenas que forman parte de un videojuego, por lo general se realiza a través de archivos externos al motor de juego. La principal tarea del módulo de carga de niveles, consiste en leer archivos de definición de niveles identificando las características específicas de cada uno, las entidades que lo componen, la forma del escenario, etc. A partir de esta lectura, se construyen, inicializan y posicionan, todos los objetos de juego que dan forma al nivel.

En caso de que al momento de la carga existiera algún nivel activo, este módulo se encargará de descargar de memoria todos los recursos que ya no sean necesarios.

El módulo de carga de niveles, posee además un mecanismo de carga anticipada de niveles y caché, para agilizar el proceso de pasaje de un nivel a otro.

- **Level Events Handler**

  Algunos niveles pueden tener eventos especiales. Desde este módulo, puede definirse para cada nivel, que ante la ejecución de determinada función global o método de un objeto en particular, se ejecute como consecuencia una acción. Por poner un ejemplo, en el nivel 5 del subgénero combate, si se mata a un armadillo utilizando las flechas, se desencadena un evento de advertencia al jugador y un posterior reinicio del nivel como penalización.

  Esta funcionalidad, también permite redefinir temporalmente métodos de las entidades de juego, para que se comporten de manera distinta en determinados niveles.

- **Texture Table Mapping**

  Consiste en un diccionario utilizado para asociar los identificadores con que se referencian los distintos assets del juego, con los nombres de archivo correspondientes. En caso de cambiar el nombre o ubicación de algún archivo, solo habrá que modificar la referencia en esta tabla.

  La organización en una tabla de referencias, permite además realizar en algunos casos una selección dinámica entre distintos archivos, según el tamaño de pantalla y nivel de definición con que se ejecute el juego.

- **Dialog System**

  Este sistema es el encargado de generar a partir de archivos de diálogos externos, un grafo con todas las posibilidades de comunicación existentes entre dos personajes determinados.

  Una de las principales tareas del sistema de diálogos, es la de guardar y controlar el flujo de ejecución de cada diálogo en ejecución. Según las necesidades, algunos nodos de texto podrán configurarse como

automáticos (avanzan transcurridos ciertos segundos), mientras que otros requerirán forzosamente una acción del jugador. En este sentido, la comunicación que permite el sistema de diálogos no se limita únicamente a la aparición lineal de párrafos de texto, sino que además permite incluir menús de opción interactivos, definición de nodos condicionales y desencadenadores de eventos.

Aún en los nodos textuales simples, el sistema controla la aparición progresiva del texto buscando imitar la cadencia del lenguaje natural, teniendo en cuenta, por ejemplo, los signos de puntuación.

- **Solid Ground**

  Si bien se estructura como una subclase de Entity, posee características particulares y constituye una base para todos los niveles del juego.

  Este módulo, específico para el desarrollo de juegos de plataformas, tiene como fin encapsular la creación de la entidad correspondiente al piso del juego, a partir de formas geométricas definidas en el editor de niveles. Estas formas podrán ser rectángulos o polígonos más complejos.

- **Background**

  Como en el caso anterior, si bien Background se estructura como una entidad, la funcionalidad que engloba esta clase motiva que se la considere en sí misma como un módulo del motor, ampliamente reutilizable en otros desarrollos.

  Las entidades que se definan como subclases de Background responderán a un efecto visual que se conoce como “parallax scrolling”. Este efecto, consiste en organizar varias imágenes en capas, de manera que las que se encuentran ubicadas más al fondo se muevan más lentamente que las ubicadas al frente, generando una ilusión de profundidad en entornos 2D [37].

  Por otra parte, las entidades que extiendan de Background tendrán la característica de repetirse automáticamente de manera horizontal.

- **Timer**

Aunque simple, esta clase permite que las entidades de juego creen tantos temporizadores como deseen, pudiendo verificar el avance del tiempo en relación a distintos eventos, accionando en consecuencia.

- **Interfaz de usuario**

  El módulo de manejo de interfaz de usuario ofrece las bases para diagramar pantallas de interfaz dinámicas y autoajustables a distintos tamaños de pantalla. La creación de estas interfaces se realiza a partir de imágenes y bloques de texto, que pueden o no tener interacción. Estos componentes podrán tener asociados además cualquier cantidad de animaciones correspondientes a alguno de estos tres tipos: de tamaño, de posición o de transparencia. Para cada animación se definirá un tiempo de inicio y fin, un par de valores de inicio y fin, y una función easing que determina la velocidad que tendrá la animación para cada uno de los instantes de tiempo.

- **Utils**

  En este módulo se organizan los métodos auxiliares de utilidad para la resolución de tareas variadas. Por ejemplo, en el caso de Raíces incluye métodos para realizar la operación de multiplicación de matrices, para generar un efecto de desenfoque sobre una imagen, y para calcular funciones easing.

5.4. Integración con herramientas externas

5.4.1. Editor de niveles

Como se ha mencionado anteriormente, la mayoría de los juegos y en particular los de plataforma, se dividen en escenas comúnmente denominadas niveles. Cada uno de estos niveles, por lo general, se define en un archivo propio que contiene toda la información específica que lo distingue del resto. Esta información puede incluir desde valores de configuración simples, hasta información sobre el terreno y de las entidades a crear, considerando en este

último caso, la posición de cada una en el escenario y sus valores de inicialización.

Para la creación de escenarios 2D orientados a videojuegos, existen distintas herramientas en el mercado. Entre las opciones de código libre más populares se encuentran Tiled Map Editor, Ogmo Editor y GLEED2D.

Luego de realizar algunas pruebas entre estos editores de escenarios 2D, se decidió utilizar **Tiled Map Editor**[15] debido a su alta funcionalidad y versatilidad. Este editor permite entre otras cosas:

- Generación de mapas de celdas (a través de las herramientas *stamp*, *fill* y *terrain*)
- Soporte para mapas ortogonales e isométricos
- Creación de objetos imagen
- Creación de figuras geométricas (rectángulos, elipses, polígonos)
- Posicionamiento a nivel de pixel de figuras geométricas y objetos imagen
- Organización en capas (ocultamiento selectivo y transparencia)

Cada uno de los niveles creados en Tiled Map Editor, puede ser guardado en formatos aceptados por juegos específicos (Droidcraft, Flare y Replica Island), en formato xml (archivo de mapa de Tiled), o en formato json. Estos dos últimos formatos, son genéricos y conservan toda la información del mapa creado, pudiendo ser leídos y procesados por cualquier aplicación de uso general. En el caso del motor que da soporte a Raíces, debido al soporte nativo provisto por el lenguaje JavaScript, se decidió trabajar con archivos .json.

[15] http://www.mapeditor.org/

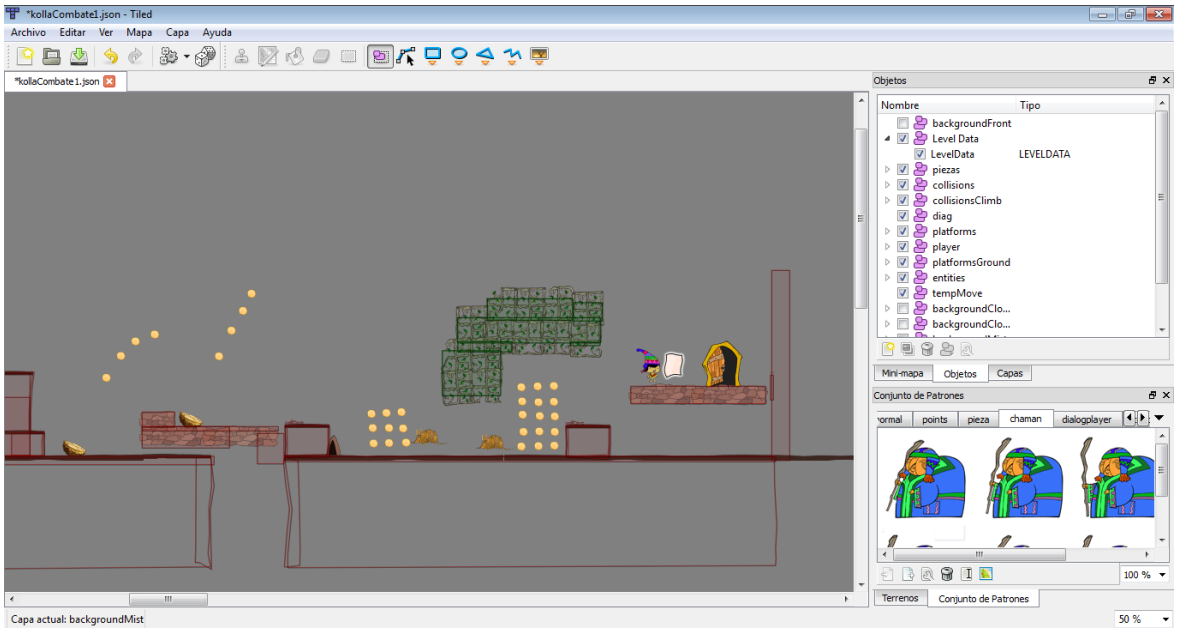


Fig. 5.5: Los niveles se diseñan a través de una interfaz gráfica intuitiva

Muchas características de Tiled están orientadas a la construcción de mapas de celdas. S. Madhav [38] define este tipo de mapas como aquellos que se componen particionando el mundo en cuadrados (u otros polígonos) de igual tamaño. Cada cuadrado referencia entonces una imagen que está en esa posición en la grilla. El contenido de cada posición generalmente es solo un número que referencia una posición dentro del conjunto de celdas utilizado. Si bien esta estrategia se utiliza en muchos juegos de plataformas, el tamaño fijo de las celdas e imágenes utilizadas, limita algunos aspectos de composición. En caso de querer crear escenarios con total libertad para la definición de los bordes, deben utilizarse otros enfoques.

Para la creación de los niveles de Raíces, se decidió prescindir de las funcionalidades de generación de mapas de celdas, para usar en su lugar, una composición de imágenes y figuras geométricas texturizadas.

En Tiled, a diferencia de lo que ocurre al utilizar celdas, las figuras geométricas y objetos imagen pueden posicionarse libremente en cualquier punto del mapa y tener información asociada (en forma de atributos clave-valor). Esta funcionalidad es utilizada intensivamente en la definición de los niveles de Raíces y permite una posterior transformación de los objetos en entidades de juego.

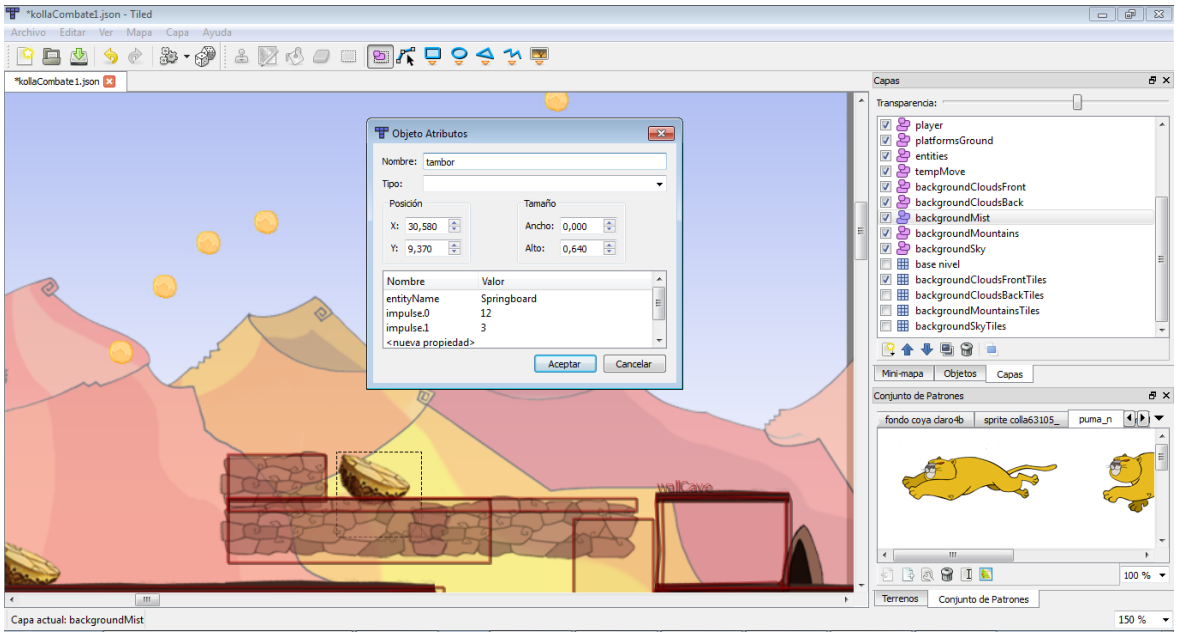


Fig. 5.6: Objetos imagen con información asociada en Tiled

El módulo de carga de niveles, espera un mapa de Tiled con los siguientes componentes:

- **Información del nivel**

  Se define como un objeto cualquiera de tipo LEVELDATA, que posee información relevante al nivel (por ejemplo: cantidad de vidas, tipo de nivel, nombre de nivel, etc.)

- **Entidades**

  Se definen como objetos imagen o figuras rectangulares, y tienen la característica de poseer el atributo clave “entityName”, una posición y un tamaño definidos.

  Opcionalmente pueden definir variables para modificar atributos simples o complejos de la entidad a crear, permitiendo una personalización de la misma.

- **Fondos**

  Se definen de igual manera que las entidades, aunque solo puede existir en el mapa una instancia de cada tipo de fondo. Esa única instancia, se replicará horizontalmente de manera automática según sea necesario.

- **Mapa de Colisiones**

Representan zonas de colisión dentro del nivel y se definen como figuras geométricas (rectangulares o poligonales).

Existen colisiones de dos tipos: *CLIMB* para zonas en donde el jugador puede trepar, y *COLLISION* para zonas sólidas. Para facilitar su identificación en Tiled Map Editor, estos objetos llevan asociados los colores verde y rojo respectivamente.

Las colisiones de tipo *COLLISION* pueden poseer, además, los atributos *ground* y *texture*, para indicar si las mismas deben ser dibujadas, y con qué textura.

Al guardar el archivo de nivel desde Tiled, toda esta información se empaqueta y queda disponible para ser leída por el motor de juego. Cuando el módulo de carga de niveles procesa este archivo identificando los distintos componentes, los datos estáticos de definición de nivel se transforman en entidades activas con las que el usuario puede interactuar.

5.4.2. Editor de diálogos

El sistema de diálogos diseñado para Raíces, permite que dos personajes entablen una conversación virtual. Esta conversación, no seguirá un único camino lineal, sino que dependerá de las elecciones realizadas por el jugador y del estado actual del juego.

Cuando se diseñan este tipo de diálogos, la especificación de los mismos a través de código puede resultar una tarea dificultosa. No solo hay que especificar para cada nodo de texto qué nodo es el siguiente, sino que hay que contemplar menús de elección, caminos condicionales, disparo de eventos en el juego, repetición y pausas en la conversación. Cada diálogo, puede modelizarse conceptualmente como un grafo con nodos de distintos tipos, con sus atributos y relaciones.

De manera similar al enfoque utilizado para incorporar niveles generados por una herramienta externa, el grafo de diálogos que representa cada conversación, también puede ser construido a través de un editor.

En este caso, la herramienta elegida fue yEd Graph Editor, un editor visual de grafos de uso general, que se ofrece de manera gratuita y permite exportar los grafos generados a archivos en formato xml.

Los grafos de diálogos creados, para permitir su posterior importación al juego, deben seguir algunas convenciones:

- **Nodos principales:** se representan con rectángulos con bordes redondeados.

  La primera letra del nombre del nodo representa su tipo (t: no jugador, p: jugador, e: evento, c: condición). Existen también nodos especiales identificados por su nombre: nodo inicio, nodo fin jugador y nodo fin no jugador.

  Para facilitar su identificación, cada tipo de nodo lleva asociado un color.

  Los atributos de cada nodo (texto, duración, repetición, avance, etc.) se representan con líneas punteadas a otros nodos.

  Para destacar el flujo de los diálogos, las líneas de avance se representan en rojo.

- **Nodos secundarios:** se representan con elipses o rectángulos con bordes rectos. Contienen los valores de los atributos.

  Además de definir el contenido del texto a mostrar, los atributos pueden indicar duración (para determinar un avance automático) y repetición (determina si el nodo debe volver a mostrarse al reiniciar la conversación)

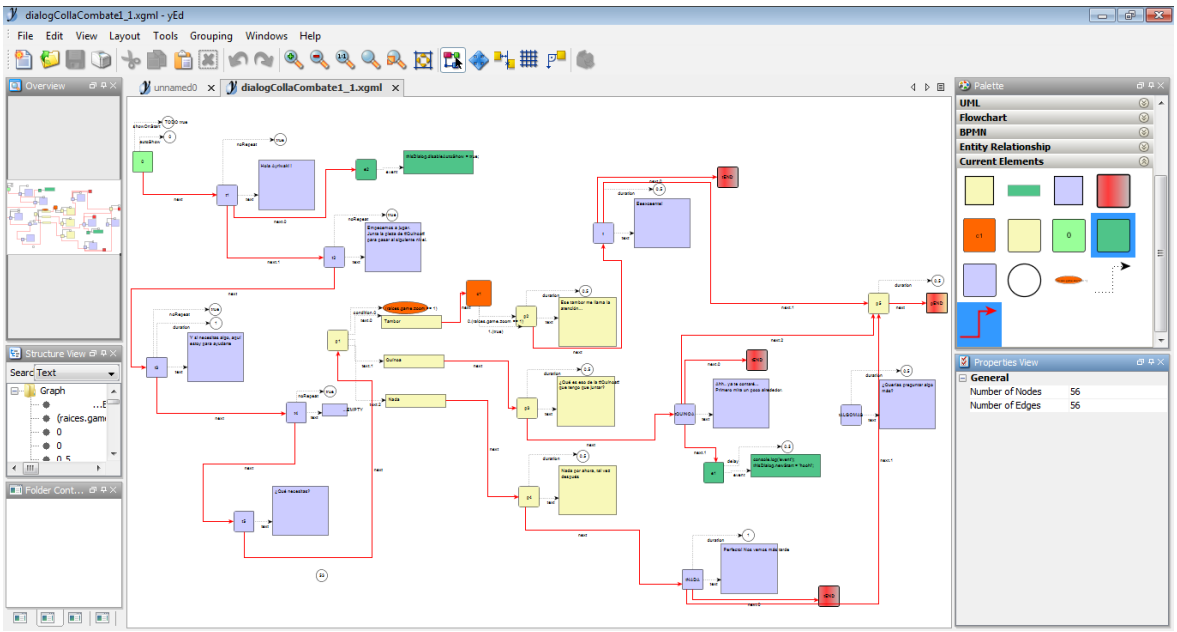


Fig. 5.7: Grafo de diálogo creado desde yEd Graph Editor

Desde la interfaz visual, pueden definirse también las siguientes características:

- **Ramificaciones condicionales:** en un momento dado, el camino de diálogo puede ramificarse según se den o no determinadas condiciones. Pueden establecerse tantas condiciones como se quiera, evaluándose las mismas una tras otra siguiendo una semántica del estilo:

  if(condición 1)

  camino 1

  else if(condición 2)

  camino 2

  else if(condición 3)

  camino 3 ...

- **Menú de elección:** ofrecen al jugador un conjunto de opciones que determinan el camino a seguir. Cada opción puede tener asociada una condición. En caso de no cumplirse esta condición, la opción no se mostrará.

- **Lanzamiento de eventos:** ciertos nodos de texto, además de mostrar diálogo, pueden desencadenar la ejecución de un fragmento de código que modifica el estado del juego. Cada evento, puede tener asociado un valor de delay, que retrasa la ejecución del código.

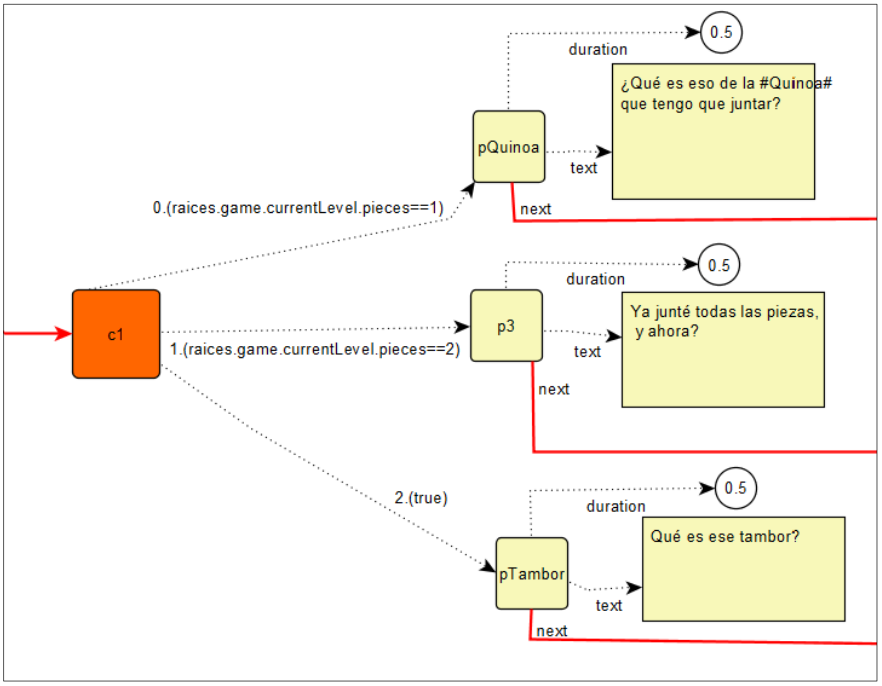


Fig. 5.8: Ramificación condicional

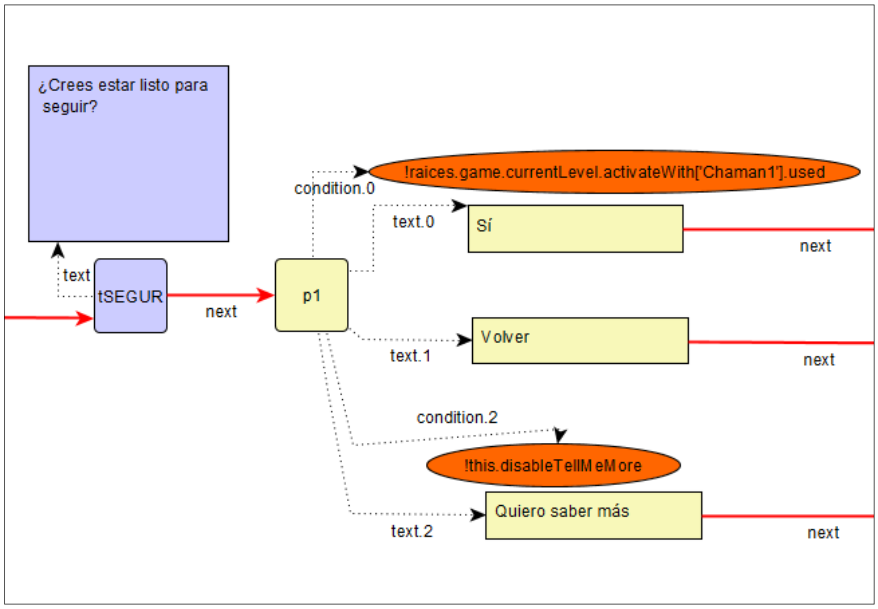


Fig. 5.9: Menú de elección con opciones condicionales

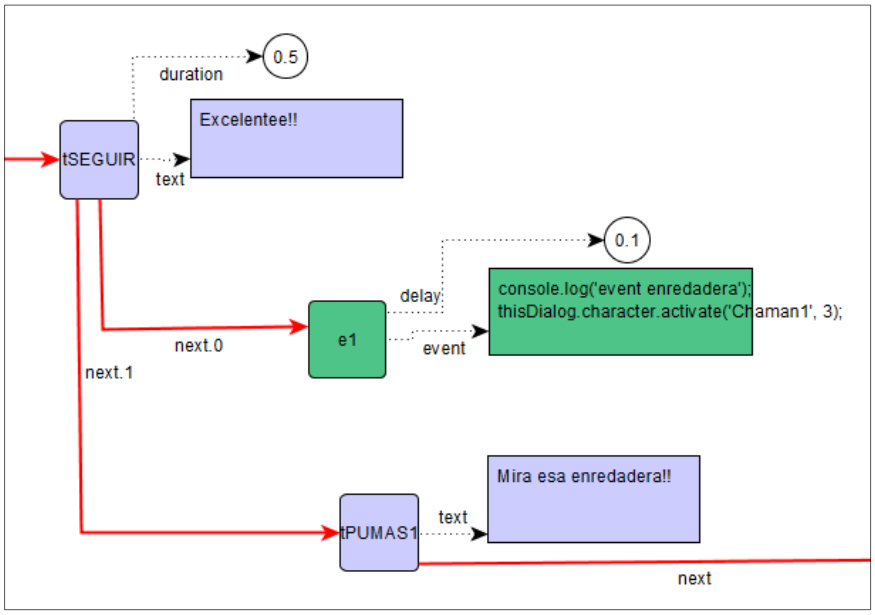


Fig. 5.10: Lanzamiento de eventos

En el momento de carga de los niveles, el sistema de diálogos de Raíces procesa los archivos xml generados desde la herramienta externa, identificando los distintos tipos de nodos y sus relaciones, para crear en memoria una estructura de datos con su representación. Esta estructura, será recorrida al momento de entablarse una conversación entre el jugador y otro personaje durante la ejecución del videojuego.

La utilización de una herramienta visual de este tipo permite una mayor simplicidad en la creación, modificación y reutilización de los diálogos.

5.5. Back-end e integración con Facebook

Raíces, como toda aplicación web que responde al modelo cliente-servidor, puede ser visto como un sistema compuesto por dos partes: una para el lado del cliente y otra para el del servidor.

Todos los aspectos de implementación de Raíces analizados hasta el momento, pertenecen al lado del cliente: tanto el motor de juego, como las entidades que dan vida al juego, se ejecutan íntegramente en un navegador web sin delegar ningún tipo de procesamiento en el servidor.

Hasta este punto, la tarea del servidor consiste únicamente en permitir el acceso y descarga de los recursos que luego se ejecutarán en el navegador (página web, código JavaScript, imágenes, sonidos, etc.). El servidor cumple las tareas de un servidor web básico.

Si bien esta funcionalidad alcanzaría para permitir el acceso a Raíces desde un navegador a través de internet, en caso de querer que el juego interactúe de alguna manera con el servidor, será necesario que el mismo incluya algún soporte adicional.

Para la implementación del back-end de Raíces, se definió la utilización de Node.js, una plataforma de programación JavaScript del lado del servidor. Esta decisión, fue motivada principalmente por la capacidad de esta plataforma para crear aplicaciones de respuesta rápida y altamente escalables, sin necesidad de utilizar un lenguaje de programación distinto al usado del lado del cliente.

Node.js, además de poseer un módulo que implementa un servidor web básico, permite incluir módulos específicos encargados de resolver todo tipo de tareas. Por ejemplo, en Raíces se utilizó el módulo Socket.IO, orientado a la creación de aplicaciones web en tiempo real, que permite una comunicación sencilla y bidireccional entre cliente y servidor sin tener que recargar la página.

Otro de los módulos incluidos en Raíces fue Node-Postgres, encargado de abstraer y facilitar el manejo de bases de datos.

A partir de estos módulos, se definió en Raíces la siguiente funcionalidad asociada al back-end:

- Responder a las peticiones de archivos a través de HTTP (servidor web)
- Guardar, a medida que se producen cambios, el progreso del jugador en una base de datos (i.e. puntaje, niveles completados, pantallas ya visitadas)
- Restaurar el progreso del jugador una vez que este vuelve a ingresar al juego luego de haberlo cerrado
- Guardar en una base de datos estadísticas de uso del juego para un análisis posterior

Por el momento, el código implementado para el manejo del servidor ha sido alojado en Heroku, plataforma de computación en la nube que soporta distintos lenguajes de programación, entre ellos Node.js, y brinda un servicio gratuito que satisface las necesidades de este proyecto: tiene soporte para HTTPS (necesario para la integración con Facebook), permite la creación y acceso a una base de datos Postgres, y está libre de publicidad.

Con respecto a la integración con la red social Facebook, la misma se realizó mediante el SDK oficial de Facebook para JavaScript[16], librería del lado del cliente que permite entre otras cosas incorporar un cuadro de Login (para que los usuarios ingresen al juego a través de su cuenta en la red social), hacer llamadas a la API de Facebook Graph[17] (para consultar datos almacenados en Facebook) y lanzar cuadros de diálogos para que los usuarios puedan realizar acciones como compartir contenido.

La integración de Raíces con Facebook es, por el momento, acotada. Como parte de esta integración se incorporó un cuadro de Login para que los jugadores ingresen al juego con su cuenta de Facebook, de manera que el progreso que realicen quede asociado a esta cuenta, y no se pierda. Además, durante el juego podrán ver su imagen de perfil cada vez que sumen puntos, y su nombre como parte de los diálogos.

[16] https://developers.facebook.com/docs/javascript/
[17] https://developers.facebook.com/docs/graph-api

5.6. Optimización

Al haber sido pensado para alcanzar un público masivo, el videojuego Raíces tuvo desde un primer momento entre sus requisitos no funcionales, el de asegurar una correcta ejecución en computadoras con recursos de hardware limitados. En particular, era de especial interés lograr que Raíces pudiera ser jugado en las netbooks del programa Conectar Igualdad.

La primera versión de Raíces construida sobre las bases de Turbulenz y Pixi.js, no satisfacía este requerimiento. El consume de recursos, tanto de memoria como de uso del procesador, que demandaba esta primera versión, excedía ampliamente los límites de estas computadoras, resultando en una pobre ejecución del juego.

En vista de estas circunstancias, empezó a aplicarse progresiva y sistemáticamente un conjunto de optimizaciones orientadas a lograr una reducción en los tiempos de actualización y renderizado visual, para obtener una ejecución más fluida del juego.

Las optimizaciones implementadas pueden ser divididas en dos grandes grupos: optimizaciones generales y optimizaciones en el renderizado.

5.6.1. Optimizaciones generales

Las optimizaciones generales son aquellas que se aplicaron para reducir el tiempo de procesamiento del motor de juego. Estas optimizaciones incluyen:

- **Refactorización del código del motor y de entidades de juego**
  Teniendo en cuenta la guía de desarrollo JavaScript presente en la documentación de Turbulenz Engine [39], se apuntó a evitar la repetición innecesaria de operaciones aritméticas, a priorizar la utilización de recorridos sobre arreglos en lugar de recorrido sobre propiedades de objetos, y a minimizar la creación de objetos, en especial dentro de loops. Esto último es importante, además, para disminuir el overhead asociado al garbage collector de JavaScript.

La identificación de las funciones u objetos que más tiempo de procesamiento consumían al momento de la ejecución, se realizó con la ayuda de la herramienta Profiling presente de manera nativa en Google Chrome. Esta herramienta permite registrar el tiempo y porcentaje consumidos por cada función, mostrando los resultados ordenados según distintos criterios.

- **Manejo de versiones**

  Se establecieron tres configuraciones basadas en el nivel de detalle con que se procesará y mostrará el juego: calidad alta (HD), calidad media (SD) y calidad baja (LQ).

  Cada una de estas configuraciones se diferencia por la inclusión o no de ciertas características (no esenciales) del juego, como por ejemplo la cantidad de capas que constituyen el fondo, la frecuencia de actualización del mundo físico y la aplicación de determinados efectos visuales.

  El manejo de versiones, como se verá más adelante, también cumple una función importante en la optimización del renderizado.

- **Actualización de entidades selectiva**

  Se agregó un mecanismo para que en cada actualización del juego, se procesen únicamente las entidades que se encuentran a determinada distancia del jugador. De esta manera, las entidades que se encuentran lejos y no son visibles, no son actualizadas, evitando procesamiento innecesario sin afectar la forma en que el jugador percibe el juego.

- **Optimización de Fondos**

  El módulo de fondos fue uno de los ejes centrales del proceso de optimización. En un primer momento, los fondos se representaban como entidades simples formadas por una imagen. Estas entidades se repetían horizontalmente, formando un conjunto que se movía de manera sincronizada. La cantidad de veces que debían repetirse estas entidades, era fijada de manera estática. Si bien este mecanismo funcionaba correctamente, para asegurar que un fondo se viera de manera completa en todas las posiciones de cámara y niveles de zoom, era necesario

establecer un número de repeticiones alto. Estas repeticiones demandaban un tiempo de procesamiento que podía ser evitado.

La optimización del módulo de fondos permitió automatizar este proceso, al encargarse dinámicamente de la creación, reutilización y eliminación de las entidades de fondo según la posición actual de la cámara y su nivel de zoom. Por otra parte, al tratar cada capa de fondo como un bloque y no como entidades separadas, el tiempo que consume actualizar las posiciones y tamaños de los fondos se redujo considerablemente.

5.6.2. Optimizaciones en el renderizado

Si bien la librería Pixi.js abstrae de manera eficiente los aspectos de más bajo nivel del proceso de renderizado gráfico, deja a criterio del desarrollador la elección de cuestiones de optimización que van más allá de este punto.

Pixi.js no incluye, por ejemplo, el concepto de cámara o ViewPort. Para generar esto, el motor de juego debe explícitamente indicar las posiciones y tamaños con que deben dibujarse cada una de las imágenes, dependiendo del nivel de zoom actual, el tamaño de la pantalla y la posición de la cámara. Al trabajar con animaciones basadas en hojas de sprites (sprite sheets, en inglés), en las cuales se tienen varios fotogramas ubicados en una misma imagen, será necesario indicar también a cada momento, qué porción de la imagen debe dibujarse. Todo este proceso es abstraído por el motor de juego, y debe ser realizado de manera eficiente. Las optimizaciones realizadas en este sentido fueron:

- **Dibujar solo entidades visibles**

  La cámara del juego, en un momento dado, solo mostrará un conjunto de entidades. Esta optimización consiste en identificar en cada iteración, antes de dibujar la escena, las entidades que se encuentran dentro del campo de visión de la cámara, marcando las restantes como no visibles. Esta marca, permite indicarle a la librería Pixi.js que la imagen no debe ser dibujada, ahorrando un tiempo considerable de procesamiento.

- **Uso de mipmaps**

Como se ha mencionado anteriormente, el uso de mipmaps permite evitar efectos visuales no deseados (es decir, aliasing), que pueden aparecer al redimensionar imágenes a tamaños más pequeños.

El uso de mipmaps, cumple por otra parte, un rol importante en la optimización del renderizado: trabajar siempre con imágenes de alta resolución, además de generar aliasing, resulta en un mayor tiempo de procesamiento. La técnica de mipmapping, al seleccionar automáticamente la resolución más acorde para cada momento, permite evitar este procesamiento innecesario.

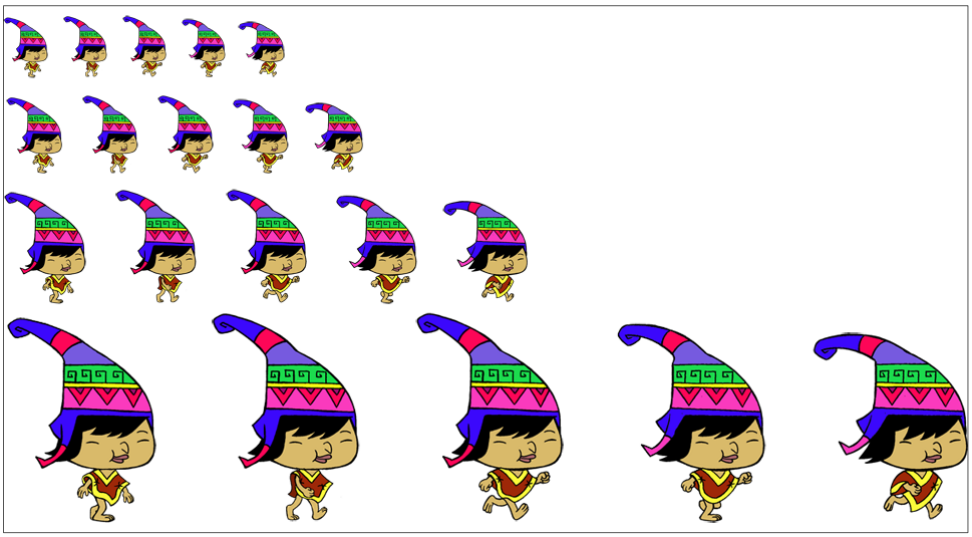


Fig. 5.11: Hoja de Sprites con mipmapping

– **Manejo de versiones**

La inclusión o no de imágenes de alta resolución, es una variable a tener en cuenta al momento de plantear una optimización. A mayor resolución, mayor será la calidad de la imagen, pero también será mayor el espacio ocupado en memoria y el tiempo que demanda su procesamiento.

Tomando las tres configuraciones de detalle planteadas en el apartado anterior (calidad alta, calidad media y calidad baja), se establecieron para cada una, límites en cuanto a la resolución máxima que tendrán las imágenes del juego:

- Calidad Baja: solo se cargan las imágenes correspondientes a las dos resoluciones más pequeñas (small y normal)

- Calidad Media: se cargan tres niveles de resolución (small, normal, big)
- Calidad alta: se cargan todas las resoluciones (small, normal, big y bigger)

– **Uso de atlas de texturas**

La técnica de utilización de atlas de texturas, busca en minimizar el overhead generado por las invocaciones de cambio de contexto de la placa de video, mediante el agrupamiento de varias texturas en un único archivo [40].

Al utilizar atlas de texturas, no solo se logra un incremento importante en la velocidad de renderizado general, sino también en el espacio de memoria ocupado.

Para el empaquetamiento de varias imágenes en una sola, se utilizó la herramienta ShoeBox[18]. Esta herramienta, se ofrece de manera gratuita e incluye varias utilidades para el manejo de imágenes, orientadas al desarrollo de juegos.

Además de ofrecer buenos resultados en la creación de atlas de texturas, los archivos generados permiten una importación directa a Pixi.js.

Para aprovechar mejor esta técnica, se crearon distintas versiones de los atlas de texturas, correspondientes a los distintos niveles de detalle con que se puede ejecutar el juego. De esta manera, se busca lograr un mejor aprovechamiento de la memoria minimizando la cantidad de texturas a ser cargadas.

[18] http://renderhjs.net/shoebox/

Fig. 5.12: Atlas de texturas

La implementación de todas estas optimizaciones, permitió ampliar el rango de dispositivos sobre los cuales puede jugarse el videojuego Raíces. Entre estos dispositivos, se cuentan las netbooks del programa Conectar Igualdad lanzadas a partir del año 2013, lo que supone una mejora significativa del rendimiento teniendo en cuenta la versión inicial. La obtención de una mayor compatibilidad más allá de este punto, se encuentra limitada por dos cuestiones: por un lado, el hecho de que el juego corra dentro de un navegador web, lo que limita el total aprovechamiento de los recursos de hardware; y por otro, la complejidad inherente del juego, el cual demanda la actualización de un sistema de física en tiempo real y el renderizado de un gran número de imágenes. Más allá de estas cuestiones, los resultados obtenidos en cuanto a rendimiento y compatibilidad son altamente satisfactorios.

6

6. Contenido y evaluación

6.1. Introducción

Como se ha mencionado anteriormente, desde un primer momento y a lo largo de todo el desarrollo, Raíces se planteó como un videojuego que lograra un equilibrio entre dos cuestiones: por un lado debía ser entretenido y atrapar a los jugadores, y por otro, debía estimular y facilitar el aprendizaje del tema pueblos originarios.
La investigación teórica realizada, contribuyó a reafirmar la postura de que para desarrollar un videojuego educativo que aproveche al máximo las posibilidades del medio, es imprescindible la comunión entre estas dos cuestiones.
Por un lado, un videojuego educativo con poco contenido, no aportará demasiado. Por otro, un videojuego con un alto grado de contenido, pero con falencias en el diseño de la jugabilidad y la integración de los mismos, probablemente no generará deseos de ser jugado, desaprovechando lo más atractivo del medio.
En este capítulo se analizará la forma en que se planteó el aspecto educativo del videojuego Raíces y las principales etapas del proceso de elaboración e introducción de este contenido como parte del juego.
También se describirán las distintas pruebas realizadas con los destinatarios de este videojuego, evaluando los resultados obtenidos tanto en los aspectos de jugabilidad, como en los pedagógicos, describiendo los cambios y decisiones derivadas de estas pruebas.

6.2. Formas de introducir contenido educativo

El aspecto "serio" de Raíces está dado por un doble objetivo: el de motivar a los niños para que se acerquen e interesen por la temática pueblos originarios, y el de enseñar contenidos específicos sobre la historia y cultura de estos pueblos.

Las encuestas realizadas en escuelas públicas y privadas de la región, revelaron que los niños conocen muy poco sobre los pueblos originarios de nuestro país, y que, por lo general, ven a sus integrantes como habitantes del pasado, que no existen en la actualidad. En las escuelas, el abordaje que se hace del tema resulta, por lo general, meramente descriptivo y superficial, fallando en estimular a los niños a interesarse por el tema con mayor profundidad.

Debido a este desconocimiento, la temática pueblos originaros, no constituye en sí misma un estímulo suficiente para atraer a los niños al uso del videojuego Raíces. En muchos casos, sumado a esto, los niños tienen una concepción negativa sobre los juegos educativos en general.

En vista de este contexto, para estimular un acercamiento de los niños, en primer lugar y sobre todo, Raíces debía ser percibido como un juego, más allá de su temática y su contenido educativo. Una vez captado el interés y la atención de los niños, el contenido se iría desprendiendo de manera natural e incremental.

En este sentido, es deseable que los contenidos a transmitir se integren directamente con las mecánicas del juego y no sean vistos por los jugadores como algo impuesto o forzado. Aun cuando no sea posible realizar una integración de contenidos a nivel de mecánicas, debería buscarse que el aprendizaje de los mismos mantenga aspectos lúdicos y sea percibido por los jugadores como parte del juego.

Al momento de definir las mecánicas de un juego educativo, deben pensarse también, al menos de manera general, las posibilidades que ofrecen esas mecánicas, y de qué formas podrían integrarse contenidos en el diseño de juego planteado.

En el caso de Raíces, la introducción de contenido educativo como parte de la mecánica principal del juego, se estructuró en distintos ejes:

- **Visual:** Este aspecto ha sido aprovechado principalmente a partir de los personajes y escenarios. Para el diseño de los personajes se sintetizaron los rasgos físicos distintivos de los pobladores originarios y teniendo en cuenta sus atuendos tradiciones. Una versión posterior del juego, podría incluir la posibilidad de personalizar el avatar, profundizando este aspecto al permitir que los niños jueguen con las distintas combinaciones y, a

través de las descripciones, descubran para qué y en qué ocasiones se utilizaba cada cosa.

Con respecto a los escenarios, cada personaje transita por espacios geográficos propios, lo que permite transmitir información sobre los lugares de origen de cada uno de los pueblos.

Por otra parte, muchos elementos que aparecen en Raíces tienen referencias visuales a las tramas y formas textiles aborígenes, componente cultural de gran importancia.

La construcción de la identidad visual (y sonora) de Raíces, fue posible gracias a la participación y el aporte de artistas de la facultad de Bellas Artes que se involucraron en el diseño de la estética del videojuego.

– **Sonoro:** La composición sonora aplicada a los videojuegos cumple la función de facilitar la comprensión del jugador, acompañar sus acciones, captar su atención, proporcionando una ambientación adecuada para lograr la inmersión del jugador en un mundo virtual, reforzando así, las emociones. Para crear la ambientación sonora de Raíces, se optó por usar sonidos evocativos al espacio y acciones del personaje.

  La música de Raíces ha sido compuesta en relación a la región y cultura del pueblo originario que participa del relato, con la presencia de instrumentos típicos mezclados con efectos electrónicos generados por MIDI (Musical Instrument Digital Interface). El espacio sonoro logrado es un híbrido de dos realidades implicadas, el universo digital en conjunto con lo autóctono nativo del país. Para la construcción sonora de los niveles correspondientes al pueblo Coya, se hace uso de instrumentos como el huayno, el charango, la caja, la quena, el sikus, el erquencho y la anata. Se pretende seguir el mismo enfoque para el resto de las culturas, permitiendo que el jugador experimente las sonoridades típicas de cada pueblo.

– **Narrativo:** Como expone G. Tavinor [41], los videojuegos claramente involucran narrativas, aunque lo hacen de una manera distinta a la presentada por las ficciones tradicionales como las novelas y las películas. En los videojuegos, existe además un componente interactivo, en el que las acciones del jugador también construyen una narración.

Si bien el género de plataformas no está fuertemente anclado en la narración, es dúctil a incorporar ciertos componentes narrativos.

El principal recurso narrativo utilizados en Raíces es el dialogo. En cada nivel, habrá un personaje (representado por un anciano o chamán) que da consejos al jugador sobre cómo seguir, brinda información sobre el significado de los distintos elementos y en ciertos niveles narra pequeñas historias.

En particular, el subgénero combate se ha diseñado para seguir un hilo histórico. A lo largo de los niveles, se van sucediendo diálogos y eventos que narran la llegada de los españoles y sus consecuencias.

Otro elemento que da continuidad a cada uno de los caminos (subgéneros) y construye narración, es la incorporación de las denominadas "piezas de nivel", elementos que deben ser conseguidos para completar los niveles, y que representan aspectos culturales significativos de cada pueblo.

La figura 6.2 presenta una captura de la pantalla que aparece al completar un nivel, en la que se muestra información correspondiente a la “pieza de nivel” conseguida.

– **Simbólico:** M. Treanor et.al. [42] analiza el potencial que tienen los videojuegos para generar “sentido” a través de las reglas, y como esto puede ser utilizado para comunicar ideas. Al hablar de “sentido”, nos referimos a la interpretación subjetiva que hace el jugador como consecuencia de la aplicación de determinada mecánica de juego.

  En Raíces, se ha intentado explotar este concepto por ejemplo en el subgénero correr, en donde la mecánica propuesta acompaña simbólicamente la búsqueda de los pueblos originarios de mantener viva su cultura e idioma frente al avance del tiempo.

  También se ha buscado promover a través de la aplicación de determinadas mecánicas el respeto por la naturaleza: si en el juego se daña la naturaleza sin sentido, esto será advertido y penalizado.

Fig. 6.1. El género Combate sigue un hilo histórico

Fig. 6.2. Pantalla con información sobre la pieza de nivel “Inti”

Además de la introducción de contenido dentro de cada nivel del juego, se decidió aprovechar también las pantallas de carga entre niveles. En estas pantallas aparecerán contenidos que se relacionan con los temas vistos en los niveles, pero que por una u otra razón, no pudieron ser incorporados de manera directa a los mismos. Cada vez que se complete un nivel, antes de avanzar al siguiente, se mostrará una pantalla inter-nivel de alguno de estos dos tipos:

- **Pantallas inter-niveles de información:**

  Las pantallas de información consisten en una imagen acompañada de uno o más párrafos de texto, que aportan datos sobre algún tema en particular. Para preservar la experiencia de usuario y mantener el dinamismo del juego, la extensión de cada párrafo se ha acotada a unas pocas líneas. Esto permite que cada uno sea leído con apenas un golpe de vista, pudiendo el jugador, en caso de interesarse por lo que se cuenta, moverse entre las distintas páginas que componen el texto completo. Esta forma de dosificación de la información, no es extraña a la gran mayoría de los jugadores, habituados, debido a la naturaleza fugaz de las redes sociales que utilizan a diario, a consumir información en pequeñas dosis.

  La imagen (foto, dibujo, o recorte periodístico) que se presente junto a los textos, servirá para aportar valiosa información visual. Además de constituir un refuerzo a lo que se cuenta, estas imágenes permitirán generar un acercamiento entre el ambiente que recrea el juego, y el mundo real.

- **Pantallas inter-niveles de trivia:**

  La palabra trivia hace referencia a un tipo de juego que consiste en responder preguntas, a partir de la elección de la respuesta correcta entre una lista de opciones disponibles.

  Este tipo de juego, existentes desde hace décadas en formato de juego de mesa, se ha ido modernizando y adaptando a distintos formatos sin perder popularidad. Un ejemplo reciente del éxito de los juegos de trivia está en Preguntados, videojuego para celulares que alcanzó más de 12 millones de descargas.

  En Raíces, este recurso ha sido introducido principalmente por dos razones: por un lado, para evaluar la existencia de un aprendizaje progresivo, y por otro, por su función como herramienta educativa en sí misma. Cada pantalla de trivia contendrá una pregunta relativa a algún contenido ya presentado en el juego, debiendo el jugador elegir una respuesta (entre cuatro opciones disponibles) dentro de un período de tiempo acotado. Aun cuando se elija incorrectamente, se mostrará la respuesta correcta con información adicional, estimulando en estos casos

un aprendizaje a partir del error. Las preguntas respondidas erróneamente, probablemente volverán a aparecer durante el transcurso del juego.

El diseño visual de estas pantallas, así como también del resto de los aspectos educativos del videojuego, se ha ido perfeccionando progresivamente en base a los comportamientos observados en los testeos y los comentarios de los jugadores.

Siempre se ha buscado que la incorporación de los contenidos educativos no resulte intrusiva y mantenga características lúdicas. En este sentido, si bien las pantallas inter-niveles pueden saltearse una vez completada la carga (del nivel siguiente), aquellos jugadores que presten mayor atención a la información y respondan correctamente las trivias, tendrán recompensas en el juego sirviendo esto de motivación.

Fig. 6.3. Pantalla Inter-nivel de información

Fig. 6.4. Pantalla Inter-nivel de trivia

Con el fin de poder identificar el funcionamiento y grado de aceptación de estas pantallas Inter-niveles, se ha desarrollado funcionalidad para la generación de estadísticas sobre su uso. Cuando un usuario interactúa con estas pantallas, el sistema captura determinadas características de esta interacción y guarda la información correspondiente en la base de datos del juego.

Para cada usuario, se mantiene un registro de la siguiente información:

- Trivias respondidas correctamente
- Trivias respondidas incorrectamente
- Trivias en las que se acabó el tiempo disponible
- Cantidad total de páginas leídas (para cada pantalla de información)
- Tiempo de permanencia (para cada pantalla de información)

El análisis de las estadísticas de uso de cada pantalla inter-nivel podría permitir identificar aquellas que no están funcionando correctamente o no generan interés, para luego realizar una reformulación de las mismas si se cree conveniente.

Por otra parte, estas estadísticas pueden utilizarse también para analizar, de manera global, el proceso de apropiación de contenidos por parte de los jugadores de Raíces.

6.3. Elección y definición de los contenidos

Como punto de partida, para identificar posibles temas de interés a tratar sobre los pueblos originarios, se realizaron reuniones con profesionales de la antropología. Estos encuentros tempranos, permitieron direccionar el enfoque que se le daría al juego y sirvió como punto de partida para la búsqueda contenidos a incluir. A partir de ellos se vislumbró la existencia de temas transversales a todos los pueblos originarios y de temas particulares para cada una de ellos, la importancia y vicisitudes de su historia, el amplio abanico de elementos que forman cada cultura, las características de su vida en la actualidad, etc.

Si bien en estas reuniones se definió la elección de los pueblos Colla, Guaraní y Mapuche como representativos de la gran variedad de pueblos originarios de Argentina, para los fines de este trabajo de tesina, se decidió acotar el diseño e implementación del juego a los niveles correspondientes al pueblo Colla.

Ya avanzado un diseño inicial de las mecánicas de juego, las reuniones con antropólogos apuntaron a la obtención de disparadores (personajes, hechos históricos o componentes culturales) que pudieran ser utilizados en el diseño particular de cada uno de los niveles.

Los antropólogos, aportaron también en distintos momentos, sugerencias de material bibliográfico y audiovisual específico, que ha servido para la elaboración del contenido educativo del juego. Entre este material, se destaca el texto “Desde adentro. Las comunidades originarias de la Argentina” [43] y la serie televisiva “Pueblos Originarios”[19].

A medida que se avanzaba en el diseño de los niveles del pueblo Colla, se fue elaborando una lista de contenidos a transmitir, agrupados en distintas categorías: Alimentación, Armas, Animales, Arte, Religión, Tradiciones, Instrumentos musicales, Historia, Actualidad y Temas Generales.

La tabla que se presenta en la Figura 6.5, enumera todos los contenidos abordados por Raíces, indicando también de qué forma se introdujeron los mismos y en qué nivel aparecen por primera vez.

Todos estos contenidos, que van desprendiéndose de manera natural y progresiva a medida que el jugador avanza y se involucra (cada vez más) con el videojuego, ofrecen una visión amplia sobre una gran variedad de cuestiones

[19] http://pueblosoriginarios.encuentro.gov.ar

relativas al pueblo Colla. Algunos jugadores, reacios a la lectura de los textos que aparecen en el juego, tal vez no alcancen un conocimiento profundo sobre todas las cuestiones, aunque es probable que adquieran al menos una visión general sobre el tema. Por el contrario, aquellos que demuestren un interés real y curiosidad por los temas planteados, encontrarán a lo largo del juego numerosas oportunidades para adquirir nuevos conocimientos.

Categoría	**Contenido a enseñar**	**Forma de introducción**	**Nivel de aparición** C: Combate R: Correr P: Pensar
Alimentación	Quínoa	Pieza de Nivel	C1
	Papa	Pieza de Nivel	C2
	Maíz	Pieza de Nivel	C3
	Terrazas de cultivo	Pieza de Nivel	C4
	Caza	Pieza de Nivel	C5
	Alimentación diaguita	Inter-nivel Información Inter-nivel Trivia	C2
Armas	Arco y flecha	Elemento de juego Inter-nivel Información	C4
	Armas diaguitas	Pieza de Nivel	C7
Animales	Armadillo	Personaje Inter-nivel Información	C1
	Puma	Personaje Diálogo Inter-nivel Información	C3
	Guanaco	Personaje	C5
	Vizcacha andina	Personaje Inter-nivel Información	P4
	Avestruz	Elemento de juego (símbolo) Inter-nivel Información Inter-nivel Trivia	R4
Arte	Los suplicantes	Elemento de juego Inter-nivel Información	P2
	Cerámica diaguita	Pieza de Nivel	C8

Religión	Chamán	Personaje Inter-nivel Información	C1, R1, P1
	Pachamama	Pieza de Nivel	P1
	Inti	Pieza de Nivel	P2
	Quilla	Pieza de Nivel	P3
	Coquena	Pieza de Nivel	P4
Tradiciones	Chicha	Pieza de Nivel	P5
	Carnaval	Pieza de Nivel	P6
	Vestimenta	Inter-nivel Información Inter-nivel Trivia	R1
	Ucu	Pieza de Nivel	R1
	Chumbi	Pieza de Nivel	R2
Instrumentos musicales	Tambor	Elemento de juego	C1, R2
	Siku	Pieza de Nivel	R3
	Quena	Pieza de Nivel	R4
	Erquencho	Pieza de Nivel	R5
Historia	Conquistador español	Personaje Inter-nivel Información	C6
	Resistencia a la conquista	Dialogo Inter-nivel Información Inter-nivel Trivia	C7
	Origen	Inter-nivel Información Inter-nivel Trivia	-
Actualidad	Cantidad de habitantes	Inter-nivel Información Inter-nivel Trivia	-
	Preservación de la cultura	Inter-nivel Información Inter-nivel Trivia	-
	Whipala	Elemento de juego Inter-nivel Información Inter-nivel Trivia	-
	Poesía Colla	Inter-nivel Información	-
Temas generales	Respeto por la naturaleza	Dialogo	C5
	Ubicación geográfica	Inter-nivel Información Inter-nivel Trivia	-
	Idioma	Inter-nivel Información Inter-nivel Trivia	-

Fig. 6.5: Contenido educativo de Raíces

6.4. Testeo de un primer Prototipo

Para obtener una primera impresión del impacto que provocaba la mecánica básica del juego, las dinámicas generadas a partir de ellas y las respuestas emocionales surgidas de la interacción con el juego, se convocó en la Facultad de Informática a niños de entre 9 y 13 años. En este primer testeo, los niños pudieron probar durante aproximadamente una hora, un prototipo, realizado en Impact.js, con los tres subgéneros implementados: Combate, Correr y Pensar.

La Fig. 6.6. muestra escenas de videos filmados durante la sesión del testeo del prototipo, donde pueden observarse que el juego provoca emociones y alto grado de concentración.

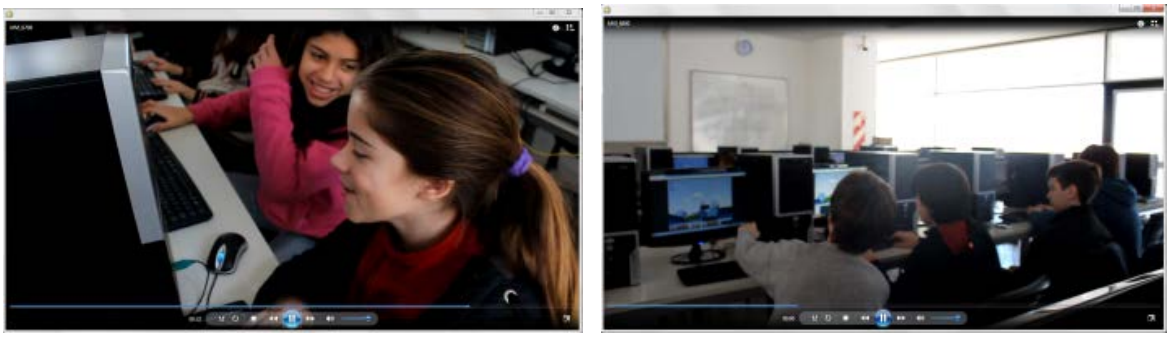
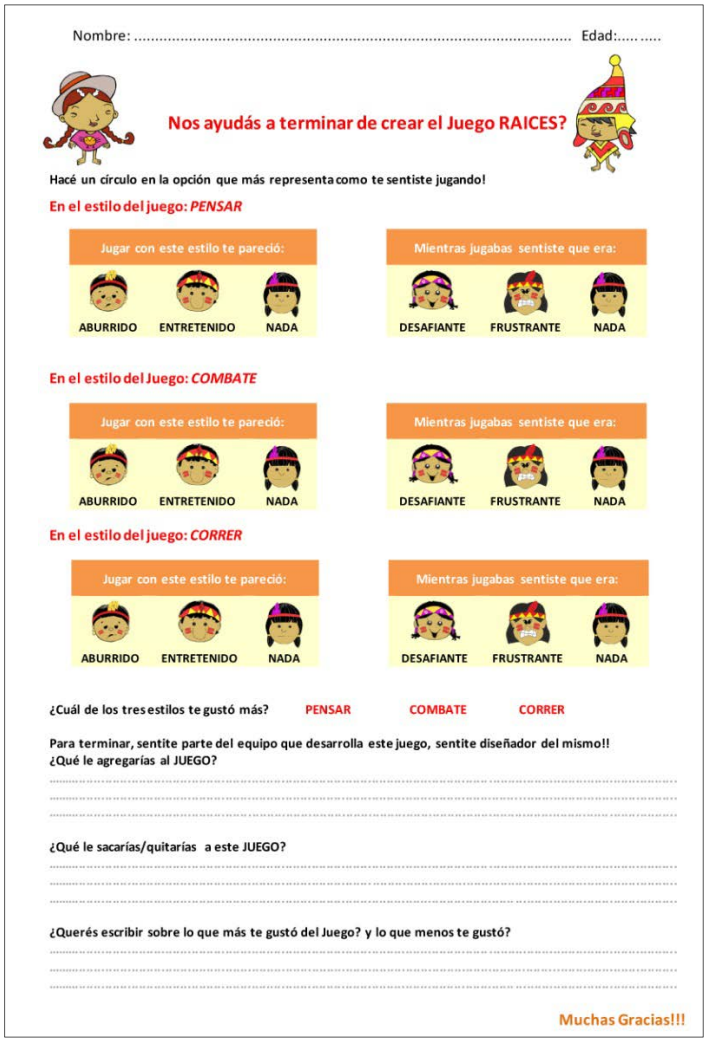


Fig. 6.6. Prueba del primer Prototipo

Se advirtió gran entusiasmo por pasar cada nivel y llegar al final del juego. Finalizado el testeo, los niños pidieron seguir jugando y se entusiasmaron con la idea de poder jugar más niveles en el futuro.

Con el fin de determinar los sentimientos provocados por los diferentes subgéneros, al finalizar la prueba se les suministró a los niños una pequeña encuesta referente a esta cuestión. La Fig. 6.7 muestra una imagen de la encuesta realizada. Como parte de los resultados de la encuesta, el 33% respondió que el subgénero Pensar fue el que más le gustó, el 45% optó por Combate y el 18% por el subgénero Correr. Si bien hay diferencias entre ellos, cada subgénero tuvo sus adeptos. La Fig. 6.8 expone otros resultados obtenidos.



Nombre: ..... Edad:.....

Nos ayudás a terminar de crear el Juego RAICES?

Hacé un círculo en la opción que más representa como te sentiste jugando!

En el estilo del juego: *PENSAR*

Jugar con este estilo te pareció:			Mientras jugabas sentiste que era:		
ABURRIDO	ENTRETENIDO	NADA	DESAFIANTE	FRUSTRANTE	NADA

En el estilo del Juego: *COMBATE*

Jugar con este estilo te pareció:			Mientras jugabas sentiste que era:		
ABURRIDO	ENTRETENIDO	NADA	DESAFIANTE	FRUSTRANTE	NADA

En el estilo del juego: *CORRER*

Jugar con este estilo te pareció:			Mientras jugabas sentiste que era:		
ABURRIDO	ENTRETENIDO	NADA	DESAFIANTE	FRUSTRANTE	NADA

¿Cuál de los tres estilos te gustó más? PENSAR COMBATE CORRER

Para terminar, sentite parte del equipo que desarrolla este juego, sentite diseñador del mismo!!
¿Qué le agregarías al JUEGO?

¿Qué le sacarías/quitarías a este JUEGO?

¿Querés escribir sobre lo que más te gustó del Juego? y lo que menos te gustó?

Muchas Gracias!!!

Fig. 6.7. Encuesta realizada a los niños luego de probar Raíces

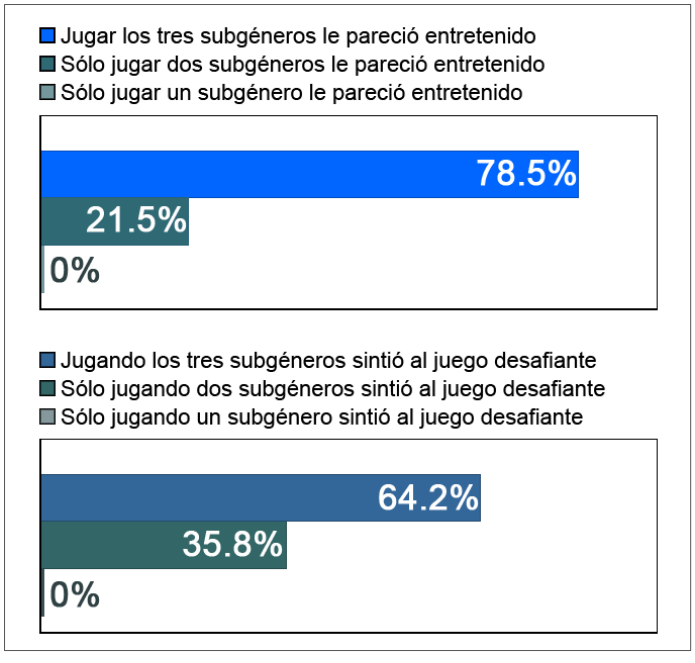


Fig. 6.8. Emociones provocadas por los subgéneros

El relevamiento de comentarios hechos por los niños y la observación in-situ de su forma de interactuar con las mecánicas planteadas, ha servido como retroalimentación en las etapas posteriores de diseño e implementación de Raíces.

6.5. Segundo testeo: primera versión con contenido educativo

Teniendo en cuenta los resultados obtenidos en el primer testeo, se avanzó con la implementación de una versión nueva de Raíces, que contaba con niveles más elaborados, ajustes en algunas mecánicas (principalmente las del subgénero correr), y la incorporación de distintos diálogos y piezas de nivel con contenido educativo.

El testeo de esta nueva versión, implementada en el motor de juego construido sobre las bases de Pixi.js y Turbulenz, se realizó en el mismo lugar y con participantes del mismo rango etario que los del testeo anterior. En este segundo testeo, al que asistieron 12 niños y niñas, además de observarse sus reacciones con el juego en sí, se prestó especial interés a su interacción con los aspectos educativos recientemente añadidos. Con respecto a esta cuestión, pudo observarse que algunos niños prestaban atención a los textos de información, mientras que otros solo hacían una lectura rápida. Esta observación motivó un posterior cambio en el diseño de la interfaz de usuario: en lugar de presentar textos largos con desplazamiento vertical, se decidió fraccionarlos en unidades más pequeñas de lectura rápida, separadas por páginas.

En cuanto al grado de entretenimiento generado por los tres subgéneros, el mismo mostró una mejoría con respecto a lo obtenido en el primer testeo, el cual ya era de por sí satisfactorio. Se notó nuevamente, a pesar de los cambios implementados, una menor preferencia por el subgénero correr. La Fig. 6.9. ilustra estos resultados.

Con respecto a las preguntas cualitativas de la encuesta, se pudo apreciar un interés de los jugadores por tener más niveles, más personajes y más escenarios. En algunos casos, destacaron también el hecho de que el juego les permitiera aprender sobre las culturas originarias.

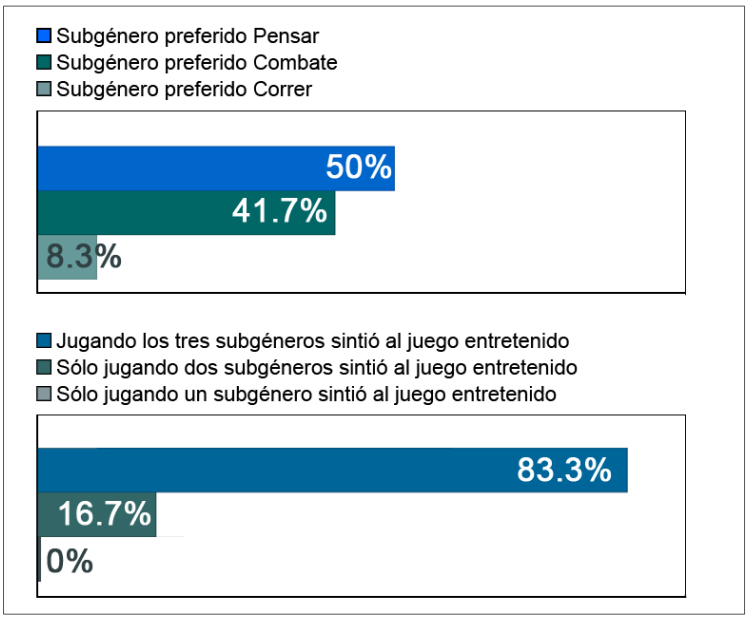


Fig. 6.9. Emociones provocadas por los subgéneros

7

7. Conclusiones y Líneas de Trabajo futuras

7.1. Conclusiones

A lo largo de este informe de tesina, se han analizado las distintas etapas involucradas en la creación del videojuego social Raíces (accesible en http://raiceseljuego.com.ar), cuyo objetivo es tornar más atractivo para los niños el aprendizaje sobre los pueblos originarios de Argentina.

Como parte de este trabajo, se analizaron los criterios que motivaron cada aspecto del diseño del juego, los motores de juego HTML5 utilizados, los aspectos técnicos concretos de la implementación de Raíces, y la integración de contenidos educativos como parte misma del juego.

Para cada una de estas etapas, se ha descripto oportunamente un marco teórico, y se han expuesto los desafíos encontrados y su forma de resolución.

La realización de un videojuego es, de por sí, una tarea compleja: debe satisfacer las expectativas de los jugadores, producir emociones, motivar el deseo de seguir jugando, etc. Para esto, además de presentar mecánicas de juego interesantes, el mismo probablemente deberá resultar atractivo en distintos aspectos: arte visual, arte sonoro, diseño narrativo, etc.

Hacer un videojuego con fines serios, suma además, otro grado de complejidad. Además de los requisitos antes mencionados, un juego serio debe buscar servir al propósito (más allá de la diversión) con el que fue concebido.

A todas las cuestiones relativas al diseño del juego, debe sumarse además la complejidad inherente de la implementación del juego diseñado, complejidad que crece al utilizar una tecnología relativamente nueva como HTML5.

La principal dificultad encontrada durante el desarrollo de Raíces se presentó con respecto a esta cuestión: para poder tener en Raíces todas las características que se deseaba tenga el juego desde su concepción inicial, fue necesario la

implementación integral de la gran mayoría de ellas, no contenidas en otro motor de juego HTML5. Por un lado, esto permitió la creación de un motor de juego propio, reusable y con una gran funcionalidad, aunque también amplió considerablemente el tiempo de implementación, retrasando la planificación original. Al momento de planificar la creación de un videojuego serio, debe tenerse en cuenta la complejidad de cada una de las etapas y la cantidad de tareas distintas involucradas en la concreción de cada una de ellas, planteando a partir de este análisis un diseño de juego acorde a los plazos y recursos disponibles.

La experiencia del trabajo realizado en Raíces y los paradigmas presentados en este informe, podrán resultar de utilidad para aquellos interesados en el diseño de juegos serios de distintos tipos.

Con respecto al juego resultante y su uso como objeto de aprendizaje, en esta tesina se propusieron distintas formas de introducción de contenido, que fueron puestas a prueba con resultados satisfactorios en diversas instancias de testeo. La integración de estos contenidos como parte del juego y no como algo forzado, fue un factor determinante para la aceptación del juego como un todo por parte de los chicos, que pudieron aprender y tomar conciencia sobre el valor de la cultura de los pueblos originarios, sin dejar de lado la diversión.

En relación a la integración de Raíces con redes sociales, si bien la misma es, por el momento, acotada, el diseño integral del videojuego se realizó teniendo en cuenta este aspecto social e incluyó características en este sentido (niveles cortos, no tiene final, pueden agregarse nuevos niveles continuamente, sistema de puntos competitivo, personalización del avatar, etc.), por lo que está preparado para que se exploten muchas otras funcionalidades provistas por este tipo de redes.

Más allá de los aspectos de Raíces que puedan seguir siendo completados y mejorados, este desarrollo ha permitido verificar principalmente dos cuestiones: desde lo técnico, las capacidades de las tecnologías HTML5 para la creación de juegos complejos embebidos en redes sociales; y desde lo educativo, la potencialidad de los videojuegos serios web como objetos de aprendizaje.

.

7.2. Líneas de Trabajo Futuras

El trabajo en esta etapa se enfocó en la creación de los primeros niveles del videojuego, centrados en los pueblos del noroeste Argentino (Collas). En total se crearon veinte niveles repartidos entre los subgéneros Combate, Correr y Pensar. Si bien esta cantidad de niveles permite la introducción de una cantidad considerable de contenido, para hacer un mayor aprovechamiento de las mecánicas ya planteadas y del motor de juego desarrollado, sería deseable tener muchos más niveles. Cuantos más niveles tenga el juego, mayor será el tiempo que se retenga al jugador, permitiendo por consiguiente profundizar determinados contenidos y enseñar otros nuevos. En particular, estos nuevos niveles podrían centrarse en las culturas e historia de pueblos originarios de otras regiones del país, dándole al juego un atractivo adicional al presentar una mayor diversidad de escenarios, músicas y personajes.

Como se ha mencionado en el apartado anterior, otro aspecto sobre el que podría trabajarse es en lo relativo a la integración con redes sociales. Para lograr un juego ciento por ciento social, Raíces podría incluir funcionalidad para seguir el progreso de otros jugadores, compartir contenido, aprovechar el sistema de puntos para desbloquear ítems de personalización, visualizar los amigos jugando actualmente y tener la posibilidad de invitar a otros nuevos, etc. Yendo un poco más lejos, este videojuego podría poseer también un modo multijugador, lo que permitiría explotar mecánicas de competencia o cooperación en tiempo real.

Otra rama de trabajo futuro, podría centrarse en la generación y análisis de estadísticas de uso del juego, con el objetivo de utilizar esta información para, por un lado, mejorar aspectos de jugabilidad (e.g. dificultad de niveles, lugares de abandono del juego, interés en determinadas mecánicas, grado de interacción social, etc.), y por otro, optimizar aspectos educativos (e.g. identificación de pantallas que generan interés, dificultad de trivias, uso de diálogos, etc.). Si bien en Raíces ya se ha planteado un primer acercamiento a la generación de estadísticas, aún puede trabajarse mucho en este sentido.

Por último y trascendiendo a la realización de Raíces, la experiencia adquirida durante esta tesina y las técnicas aplicadas durante su desarrollo, abren también otras líneas de trabajo futuro: la formulación y concreción de nuevos proyectos, con temáticas y objetivos diferentes, también relacionados con el cada vez más rico universo de los juegos serios.

Referencias

[1] [Online] Encuesta Nacional de Consumos Culturales http://sinca.cultura.gob.ar/sic/publicaciones/libros/EECC.pdf (último acceso 1 de Julio de 2015)

[2] Gandulfo De Granato, M. A., Taulamet, M., & Lafont, E. (2004). El juego en el proceso de aprendizaje. Buenos Aires, La Crujía, 3.

[3] Salen, K., & Zimmerman, E. (2004). Rules of play: Game design fundamentals. MIT press. ISBN: 978-0262240451

[4] Juul, J. (2010). The game, the player, the world: Looking for a heart of gameness. PLURAIS-Revista Multidisciplinar da UNEB, 1(2).

[5] Djaouti, D., Alvarez, J., Jessel, J. P., & Rampnoux, O. (2011). Origins of serious games. In Serious games and edutainment applications (pp. 25-43). Springer London.

[6] Abt, C.C. (1970) Serious Games. Viking Press, New York. ISBN: 978-0819161482

[7] Michael, D. R., & Chen, S. L. (2005). Serious games: Games that educate, train, and inform. Muska & Lipman/Premier-Trade. ISBN: 978-1592006229

[8] Blunt, R. (2007). Does game-based learning work? Results from three recent studies. In Proceedings of the Interservice/Industry Training, Simulation, & Education Conference (pp. 945-955).

[9] Gouveia, D., Lopes, D., De Carvalho, C. V., & Batista, R. (2012, March). Time Mesh: An Educational Historical Game. In Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), 2012 IEEE Fourth International Conference on(pp. 171-173). IEEE.

[10] Anderson, E. F., McLoughlin, L., Liarokapis, F., Peters, C., Petridis, P., & Freitas, S. (2009). Serious games in cultural heritage.

[11] Huang, C. H., & Huang, Y. T. (2013). An annales school-based serious game creation framework for taiwanese indigenous cultural heritage. Journal on Computing and Cultural Heritage (JOCCH), 6(2), 9.

[12] Knol, E., & De Vries, P. W. (2011). EnerCities-A serious game to stimulate sustainability and energy conservation: Preliminary results. eLearning Papers, (25).

[13] Bogost, I. (2005). Videogames and the future of education. On the Horizon,13(2), 119-125.

[14] Gee, J. P. (2003). What video games have to teach us about learning and literacy. Computers in Entertainment (CIE), 1(1), 20-20.

[15] Esnaola Horacek, G. et al. (2014). Videojuegos en la educación. Aularia, 3(1) Enero. pp: 21-26.

[16] Zyda, M. (2005). From visual simulation to virtual reality to games. Computer,38(9), 25-32.

[17] Rieber, L. (1996). Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. Educational Technology Research & Development, 44 (2), 43-58.

[18] [Online] Bogost: Let's make 'earnest' games, not 'serious games' http://www.gamasutra.com/view/news/194490/Bogost_Lets_make_earnest_games_not_serious_games.php (último acceso 1 de Julio de 2015)

[19] Fullerton, T., Swain, C., & Hoffman, S. (2004). Game design workshop: Designing, prototyping, & playtesting games. CRC Press. ISBN: 978-1578202225

[20] Novaro, G. (2002). Indios,“aborígenes” y “pueblos originarios”. Sobre el cambio de conceptos y la continuidad de las concepciones escolares. VII Jornadas Regionales de Investigación en Humanidades y Ciencias Sociales.

[21] Nygren, N., Denzinger, J., Stephenson, B., & Aycock, J. (2011, August). User-preference-based automated level generation for platform games. InComputational Intelligence and Games (CIG), 2011 IEEE Conference on (pp. 55-62). IEEE.

[22] Fromme, J. (2003). Computer games as a part of children’s culture. Game studies, 3(1), 49-62.

[23] Lazzaro, N. (2004). Why we play games: Four keys to more emotion without story.

[24] Järvinen, A. (2009, August). Game design for social networks: interaction design for playful dispositions. In Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games (pp. 95-102). ACM.

[25] Esnaola Horacek, G., & Francisco, R. D. (2013). Videojuegos en redes sociales. Nuevas perspectivas en edutainment. Edit Laertes. Barcelona. ISBN: 978-8475849133

[26] Brathwaite, B. (2007). Facebook's Parking Wars' Play Dynamics–Updated. Applied Game Design.

[27] Hunicke, R., LeBlanc, M., & Zubek, R. (2004, July). MDA: A formal approach to game design and game research. In Proceedings of the AAAI Workshop on Challenges in Game AI (Vol. 4).

[28] Lewis, M., & Jacobson, J. (2002). Game engines in scientific research. Communications of the ACM, 45(1), 27-31.

[29] Anderson, E. F., Engel, S., Comninos, P., & McLoughlin, L. (2008, November). The case for research in game engine architecture. In Proceedings of the 2008 Conference on Future Play: Research, Play, Share (pp. 228-231). ACM.

[30] Thorn, A. (2011). Game engine design and implementation. Jones & Bartlett Publishers. ISBN: 978-0763784515

[31] van Niekerk, A.J. (2006) The Strategic Management of Media Assets; A Methodological Approach. Allied Academies, New Orleans Congress, 2006.

[32] [Online] HTML5 Game Engines Wiki https://github.com/bebraw/jswiki/wiki/Game-Engines (último acceso 1 de Julio de 2015)

[33] [Online] HTML5 Game Engines List http://html5gameengine.com (último acceso 1 de Julio de 2015)

[34] Gregory, J. (2009). Game engine architecture. CRC Press. ISBN: 978-1568814131

[35] Rabin, S. (Ed.). (2010). Introduction to game development. Cengage Learning. ISBN: 978-1584506799

[36] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: elements of reusable object-oriented software. Pearson Education. ISBN: 978-0201633610

[37] Van der Spuy, R. (2012). Foundation Game Design with HTML5 and JavaScript. Apress. ISBN: 978-1430247166

[38] Madhav, S. (2013). Game Programming Algorithms and Techniques: A Platform-Agnostic Approach. Addison-Wesley Professional. ISBN: 978-0321940155

[39] [Online] Turbulenz JavaScript Development Guide, Performance Techniques http://docs.turbulenz.com/js_development_guide.html#performance-techniques (último acceso 1 de Julio de 2015)

[40] Wloka, M. (2005). Improved batching via texture atlases. Shader X3: Advanced Rendering with DirectX and OpenGL, 155-167.

[41] Tavinor, G. (2009). Videogames and Narrative. The Art of Videogames, 110-129. ISBN: 978-1405187886

[42] Treanor, M., Schweizer, B., Bogost, I., & Mateas, M. (2011, June). Proceduralist Readings: How to find meaning in games with graphical logics. InProceedings of the 6th International Conference on Foundations of Digital Games (pp. 115-122). ACM.

[43] Martins, M.E. (2009) Desde adentro. Las comunidades originarias de la Argentina. - 1a ed. - Buenos Aires: Fundación de Historia Natural Félix de Azara: Ministerio de Educación de la Nación. ISBN: 978-9872354558