Introducción a la Programación de Videojuegos

Trabajo Práctico Integrador Grupal
“BOSS”

Universidad Nacional de Quilmes
2017

Contenido

Introducción

La programación de videojuegos es una actividad eminentemente práctica. A su vez, requiere mucho ingenio y creatividad, dado que programar cada juego es distinto y los desafíos técnicos que se presentan varían mucho en función de lo que se quiera construir.

El presente trabajo práctico tiene por objetivo que los alumnos ejerciten las técnicas y conceptos mostrados durante las clases y, a su vez, que cada grupo realice el proceso completo de la construcción de un videojuego, desde la concepción de la idea hasta la implementación y demostración.

La experiencia de cada grupo será distinta en función de la tecnología elegida y el juego que se quiera construir. Se busca que cada grupo transmita la experiencia adquirida en este ejercicio luego de terminado el trabajo.

Así como el *Boss* de un videojuego, el presente trabajo es un desafío, el más grande de la materia, y que debe superarse con una mezcla de habilidad, estrategia y tenacidad.

Consigna

Cada grupo deberá realizar las siguientes tareas:

- **Concebir al menos 3 ideas de videojuegos que deseen construir**. Estas ideas serán validadas con los docentes para determinar si son adecuadas. Si al cabo de cierto plazo el grupo no concibiera una idea adecuada, los docentes propondrán replicar un juego existente.
- **Evaluar la tecnología a utilizar y seleccionarla.** El grupo podrá elegir en qué tecnología trabajar para la construcción del videojuego. Cada propuesta podrá tener una tecnología distinta, y hasta cierto punto será posible cambiar la decisión. Más avanzado el cuatrimestre, la decisión de la tecnología no será reversible. Los docentes evaluarán si la elección de la tecnología es adecuada para los objetivos académicos.
- **Construir el juego, aplicando un proceso de desarrollo.** La mayor parte del cuatrimestre, los alumnos trabajarán en la construcción propiamente dicha del juego. Esto implicará programar la lógica del juego y las herramientas necesarias para ello, así como también la construcción o recopilación de un contenido y arte mínimo para que el juego funcione. También deberán planificar qué prestaciones desean incorporar al juego, y rastrear el progreso. Si bien el contenido de las clases ayudará a esto, se recomienda que los alumnos no esperen a recibir todo el contenido antes de ponerse a desarrollar. Hacer juegos lleva mucho más tiempo de lo que parece.
- **Presentar el juego y realizar retrospectiva.** Al finalizar la cursada, cada grupo expondrá el juego que construyó. Mostrará aquellos aspectos que considere más valiosos, y comentará la experiencia adquirida, problemas encontrados y cómo fueron resueltos. A su vez, evaluará las decisiones más importantes, incluyendo la decisión de la tecnología. Finalmente, comentará cuáles serían los próximos pasos si se continuara con el juego.

A continuación se detalla para cada tarea algunos tips y expectativas, con el objetivo de ayudar a los grupos a realizarlas

Concepción de la idea

“Ideas are cheap. Execution is everything” - Chris Sacca

Existe una tendencia a sobrevalorar las ideas. A considerar que el éxito viene de una muy buena idea que algún genio tuvo. A creer que las ideas que uno tiene no son tan buenas, y que por lo tanto tienen poco valor.

La experiencia y el trabajo demuestra que la mayoría de las ideas son “baratas”, y así como son “baratas”, tienen poco valor por sí mismas. Una idea en el vacío no tiene defectos ni problemas. Recién una vez que se intenta llevarlas a la práctica es que uno las termina de conocer de verdad. La realidad es que es muy fácil concebir ideas en comparación con el tiempo que lleva ponerlas en práctica.

Teniendo en cuenta lo anterior, se busca que cada grupo decida qué tipo de juego quiere construir. No es necesario que sea original, ni tampoco innovador, pero debe ser una idea que pueda ser llevada a la práctica, y sobre la cual se pueda iterar.

Cada grupo deberá preparar al menos tres propuestas de trabajo práctico. Para concebir estas propuestas, se recomienda no censurar las ideas propias. Listar cualquier idea, por más mala que parezca y recién después de armar una lista de buen tamaño, seleccionar aquellas que se consideren mejores. La intención es aplicar la técnica de *brainstorming*.

Para seleccionar las ideas (después de generarlas), se recomienda considerar los siguientes factores:

- **Limitaciones tecnológicas:** Los juegos que se construirán serán 2D, y difícilmente puedan ser técnicamente superiores a juegos existentes
- **Alcance:** Considerar que el tiempo del cuatrimestre es acotado. Hacer juegos lleva tiempo. Lo ideal es que la idea permita iteraciones: Primero construir una parte y ya tener un juego funcionando, y luego extenderse de cierta manera. Así, es posible tener una idea quizás ambiciosa, pero con un resultado parcial que resulte satisfactorio.
- **Appeal:** La idea debe resultar atractiva al menos para los miembros del grupo. De esta manera, cada integrante se sentirá motivado para realizar más trabajo, y el resultado será mejor.
- **Material existente:** Dado que estamos aprendiendo, puede ser más realizable y más aprovechable alguna idea para la cual ya existan ejemplos y material existentes. Hacer un juego que se adapte a un género popular (platformer, estrategia, rpg) ayudará a encontrar material más fácilmente.

Comunicación de la idea

Luego de seleccionadas las ideas, deberán elaborarse a un punto tal para poder comunicarlas adecuadamente. Cada idea debe detallarse en un documento no muy extenso, pero que contenga al menos los siguientes puntos:

- **Nombre del juego**
- **Género(s):** Una lista de los géneros con los que podría encajar el juego, entendiendo género como un patrón de comunicación. Por ejemplo, Platformer con elementos de RPG. Si el juego no encajara en ningún género de manera adecuada, podrá omitirse
- **Concepto del juego:** Breve descripción de en qué consiste. Mecánicas principales. Flujo principal, es decir, qué es lo que tiene que hacer el jugador normalmente. En *Megaman* por ejemplo el flujo principal sería: Explorar el nivel, superar obstáculos saltando y disparando, matar enemigos saltando y disparando, recolectar energía y upgrades, vencer el boss.
- **Juegos similares:** De existir, juegos que se consideren similares, o que se consideren "inspiración" para este juego en particular.
- **Motivación del jugador:** ¿Qué es lo que haría atractivo al juego para los jugadores? Intentar ser lo más específico posible. No decir que va a ser "divertido", sino explicar qué experimentará el jugador que lo hará querer jugar. Por ejemplo, en *Megaman* la motivación del jugador podrá ser sentirse hábil y poderoso al superar obstáculos difíciles, y al ser cada vez mejor en el juego, así como también disfrutar de la historia del juego.
- **Tecnología tentativa:** De realizar esta idea, en qué tecnología se haría.

Este documento se utilizará para comunicar la idea y para asegurarse de que haya cierto consenso acerca de en qué consiste entre los miembros del grupo.

Elección y evaluación de la tecnología

La tecnología elegida condicionará mucho cómo se construirá el juego. La decisión debería tomarse en función de diversos factores, pero principalmente deberá tenerse en cuenta que el objetivo es **aprender a programar juegos**. Esto implica:

- Es preferible una tecnología sencilla y fácil de aprender a una poderosa
- Es preferible una tecnología orientada a la programación a una orientada al diseño o la parte gráfica
- La tecnología a utilizar debe permitir aplicar buenas prácticas de programación

A su vez, habrá ciertas restricciones en la tecnología a elegir:

- **Tiene que ser 2D**. 3D tiene matemática mucho más compleja, y requiere también contenido más rico habitualmente, y más difícil de elaborar.
- **Tiene que tener programación**. No vale elegir un motor tipo RPG engine y sólo usarlo como editor de escenarios / niveles
- **Tiene que ser gratuita al menos para un proyecto educativo**. La idea es que ustedes puedan usar el juego que hagan como parte de su portfolio, y se complica si piratean el software.

A continuación también se listan las tecnologías recomendadas por la cátedra:

- **LibGDX**: Tecnología oficial. Multiplataforma. Profesional. Poderosa. Librería de bajo nivel. Difícilmente se la pueda llamar motor. Combinada con Ashley, provee una estructura mínima para el juego.
- **Unity**: Motor muy poderoso y muy usado en la industria. Muy bueno para prototipar. Se puede complicar para llevar acabo ciertas tareas, como generación procedural.
- **Godot**: Motor programado por un argentino. Muy completo y poderoso. Programable en varios lenguajes.
- **Phaser**: Framework para programar juegos que corran en el navegador, usando HTML5.
- **PyGame**
- **HaxeFlixel**

Para elegir una tecnología, también, se recomienda realizar un prototipo. **Es ideal haber hecho funcionar al menos un pseudo juego en una tecnología antes de comprometerse a hacer el BOSS en ella.**

Es posible cambiar la tecnología elegida hasta la definición del scope, el 6 de septiembre.

Construcción del Juego

Una vez seleccionada la idea y la tecnología en la cual se llevará adelante el proyecto, es momento de construirlo. El objetivo es que el grupo se organice de manera tal de poder realizar las tareas de manera eficiente y con la participación de todos los miembros.

Material de Soporte

Dado que el juego a construir depende mucho de las decisiones tomadas por el grupo, es importante que los miembros se tomen el tiempo para definir alcance y organizar el trabajo. Se recomienda elaborar lo siguiente:

- Documento de notas / discusión / grupo de discusión: El objetivo es tener un repositorio de notas y decisiones, que permita que el grupo discuta y elabore cómo debería ser el juego. Una alternativa es un **documento de google**, usando comentarios, pero cualquier herramienta que permita escribir texto libre y discutir sobre eso puede servir
- Herramienta de definición de alcance y trackeo del progreso. El objetivo es poder tener una idea de qué es lo que se hizo, que es lo que hay que hacer en el corto plazo, y qué es lo que hay que hacer en el largo plazo. De yapa, puede saberse también quién está haciendo qué. **Trello** es una excelente herramienta para esto.
- Prototipos/pruebas de concepto: Cada grupo usa una tecnología de su elección, que puede o no ser nueva para algunos o todos los miembros. Usualmente eso constituye un obstáculo para poder desarrollar el juego, dado que es necesario conocer la tecnología que se usa para poder implementar gran parte de las funcionalidades. Se recomienda que al menos uno de los miembros del grupo dedique parte de su tiempo a elaborar prototipos y pruebas de concepto, a averiguar cómo realizar cierta funcionalidad en la tecnología elegida. Estas construcciones deberían tener algo que ver con el juego a desarrollar: Deberían o bien ser una versión mínima del juego en cuestión, o probar alguna mecánica o parte del juego, para luego incorporarla.

Poner en marcha la rueda

La parte más importante del trabajo práctico es el código y los assets del juego en sí. Es por esto que es primordial comenzar a construir el juego lo más rápido posible, y atravesar las primeras barreras para todos los miembros del grupo.

En este sentido, se busca que en las primeras semanas los alumnos se dediquen a completar un flujo de desarrollo y de esta manera poner en marcha la maquinaria. Para esto, se recomienda:

- Todos los miembros del grupo deben instalar y probar la tecnología elegida. Esto implica construir una aplicación básica, un juego sencillo que puede ser un tutorial
- Uno de los miembros del grupo debe crear el proyecto que más adelante será la pieza final del TP, el código que se entregará. Luego, debe compartir este código a través de un SCM, que puede ser Git, Mercurial o SVN. GitHub es una buena opción. Todos los otros miembros del grupo deben poder obtener el proyecto, compilarlo, correrlo y realizar una contribución mínima, que puede ser agregar un método, un comentario, o un asset. De esta manera el grupo se asegurará que todos sus miembros están en condiciones de realizar contribuciones.

Hitos, priorización, progreso y alcance

“I have yet to see any problem, however complicated, which, when you looked at it in the right way, did not become still more complicated.” - Poul Anderson

A la hora de construir software en general, y videojuegos en particular, existe una tendencia a definir de una manera muy abierta el alcance. Esa tendencia a su vez lleva a subestimar el trabajo que lleva construir una pieza de software en particular.

Al realizar este trabajo práctico, y debido a que el alcance es definido por el grupo, es muy probable que el grupo se encuentre en una situación en similar a lo descripto más arriba. Es por esto que es importante tomar medidas para impedir una paralización, y que el grupo pueda hacer el mejor trabajo posible.

Hitos

La parte más importante es definir **hitos**. Un hito será un estado particular del proyecto que se considerará importante, un estado que implicará que una parte importante del juego estará implementado.

La implementación del juego deberá dividirse en hitos. Cada hito incrementará funcionalidad sobre el hito anterior, y se supone que la importancia de los hitos va en orden decreciente. No es necesario, sin embargo, haber terminado un hito para comenzar a trabajar sobre el siguiente. Sí se recomienda no trabajar en demasiados hitos en paralelo, porque eso haría que pierdan sentido.

El primer hito será el más importante, porque arrojará mucha información y mucho aprendizaje para los miembros del grupo. Permitirá determinar qué tan realista es el alcance definido, y si el concepto del juego funcionará como se espera.

Se recomienda tener en cuenta las siguientes características para definir el primer hito:

- **Debe capturar la esencia del juego, pero de una manera minimalista.** Esto implica seleccionar las mecánicas más importantes, y las características del mundo más representativas, e incluirlas en este hito, en su versión más simple. Por ejemplo, en Megaman la esencia del juego sería un personaje que pueda saltar por plataformas y dispararle al menos un enemigo.
- **Debe ser lo menos ambicioso posible.** El primer hito tiene que completarse rápidamente, idealmente para la fecha de definición del alcance el **6 de septiembre**. Si el hito es muy ambicioso se dilata la concreción del mismo y, por lo tanto, se atrasa la información y conocimiento que aporta.
- Es preferible que abarque mucho sin tanta profundidad, a que abarque poco de manera muy profunda. Es decir, no enfocarse en los detalles, sino más bien preocuparse en construir algo que sea “jugable” y que capture la esencia del juego.

El primer hito está relacionado con el concepto de Mínimum Viable Product (https://en.wikipedia.org/wiki/Minimum_viable_product), si bien no coincide exactamente dado que es más bien un producto incompleto.

Alcance y Priorización

Como parte del trabajo práctico, **los alumnos deberán definir un alcance** para el juego, que se intentará completar para el final del cuatrimestre. Este alcance deberá estar detallado a través del Trello, o la herramienta que se elija.

Este alcance, sin embargo, será **dinámico**. Esto implica que a medida que vaya avanzando el proyecto y se tenga más información del juego y nuevas ideas podrán agregarse nuevas funcionalidades a implementar.

La idea es mantener una lista priorizada de las funcionalidades / tareas a implementar. De esta manera, siempre se sabe con qué parte continuar el desarrollo, a pesar de que el alcance puede estar variando. Al menos uno de los miembros del grupo debería encargarse de mantener actualizado el alcance, es decir, reflejar las decisiones que se fueron tomando en la herramienta que se use para trackearlo.

Dividir y conquistar: Al principio, muy probablemente los ítems de alcance que se especifiquen serán muy amplios y algo ambiguos. Es muy difícil marcar como finalizados esos ítems, y es por ello que es una buena idea dividirlos, para que los incrementos sean más pequeños y haya una sensación de progreso.

Por ejemplo, un ítem de alcance podría ser “Implementar animaciones del personaje principal”. Es un ítem quizás muy grande, por lo cual se puede ir dividiendo en varios ítems, como ser “Integrar animación Idle al personaje principal”, “Integrar animación de caminar al personaje principal”, etc. Como regla general, ítems más pequeños son siempre más manejables.

Prioridad: Para mantener el foco, el grupo deberá asegurarse de estar trabajando en lo más importante. ¿Y cómo determinar lo más importante? El primer criterio debiera ser “aquello que hará que mi programa sea un juego”. Es decir, no enfocarse en detalles hasta que no esté completada la “jugabilidad” principal, el flujo principal del juego.

Una vez terminado ese flujo, el criterio para la prioridad depende de cada juego y deberá determinarse entre los miembros del grupo y con ayuda de los docentes. Algunos juegos requerirán hacer foco en programar nuevas mecánicas. Otros requerirán generación de contenido.

Checkpoint

El checkpoint es la primera entrega del trabajo práctico. El objetivo es verificar que el grupo esté encaminado hacia realizar un juego de una calidad adecuada, y de paso consistir en el primer punto de evaluación del BOSS.

El checkpoint coincide con una clase “taller” que permitirá realizar progreso del lado de los alumnos y observar al grupo trabajar del lado de los docentes.

Por cada grupo, también, los docentes evaluarán la calidad del trabajo realizado y el avance logrado. El objetivo es que haya mínimamente un producto que pueda considerarse un juego, al cual pueden faltarle mecánicas, contenido y pulido, pero que debe poder “jugarse”.

Como guía, se tendrán en cuenta los siguientes criterios:

- Input del usuario: El juego deberá permitir al jugador realizar gran parte de las acciones que se espera estén en el producto final.
- Integración correcta de parte de los gráficos.
- Implementadas gran parte de las mecánicas que se espera que tenga el producto final, intentando cubrir variedad.
- Herramientas para construcción de contenido.
- Implementación de estructura de “nivel” o dosificación del contenido.
- Justificación de las decisiones de diseño tomadas.
- Grado de planificación.
- Modularización del código.

La evaluación será en base a anotaciones que los docentes tomarán durante la clase de taller, teniendo en cuenta los puntos anteriores y quizás otros criterios. No hace falta prepararse especialmente, sino estar listo para trabajar durante la clase.

Entrega Final (Final Showdown)

Luego de un cuatrimestre entero de trabajo, es momento de demostrar el trabajo realizado, a todo o nada.

La entrega final del trabajo práctico consistirá en una presentación de 40 minutos de duración aproximadamente que cada grupo realizará frente al resto del curso. Esta presentación estará dividida en dos partes principales:

- **Demo:** Consistirá en mostrar el juego, intentando exaltar sus mejores características y mostrando la mayor variedad de contenido posible. Tendrá carácter de espectáculo, por lo que la idea es hacerla lo más entretenido posible. Se busca emular las demos de juegos que se hacen en las convenciones de videojuegos, como E3. **Deberá durar como máximo 10 minutos.**
- **Exposición**: Será una presentación más académica y técnica. La intención es que cada grupo pueda compartir su experiencia desarrollando el juego para que los otros grupos también puedan enriquecerse un poco con ella, y de paso otorgar material para que los docentes puedan evaluar el trabajo práctico.

A continuación se listan algunos punto y observaciones para cada una de estas dos partes:

Demo (10 minutos máximo)

“Gentlemen, you had my curiosity… but now you have my attention.”

El objetivo de la demo es simple: Apelar a los intereses y emociones de quienes la observan para que quieran jugar tu juego. Se busca entonces demostrar y hacer énfasis en los aspectos más interesantes del juego, y presentarlos de una manera espectacular.

Ahora bien, el qué es sencillo. El cómo, en cambio, es más difícil. Sin intención de tener una respuesta definitiva, a continuación se listan una serie de consideraciones y recomendaciones a tener en cuenta para la demo:

- **Conoce a tu público:** Las personas no son todas iguales. La demo debería adaptarse al público objetivo de tu juego en general, y al público que presencia tu demo en particular. Por ejemplo, si el público tiene cultura de videojuegos, las referencias a videojuegos más antiguos tendrán muchísimo impacto. Distinto será si es un público más casual.
- **Conoce se mostrará:** La idea es comenzar a armar la demo listando a grandes rasgos qué se quiere mostrar del juego. Qué mecánicas, qué interacciones, qué contenido.
- **Arma un script:** Saber qué se quiere mostrar no es suficiente para lograr una demo fluida e interesante. El cómo también es importante, y eso se detalla en el script: Una serie de acciones que realizarán los jugadores y el juego para mostrar lo que se listó previamente, en un orden determinado. Es ideal también que el script siga una curva de interés, obteniendo la atención de los espectadores, y conduciéndolos hacia un clímax quizás una resolución, o quizás un cliffhanger prometiendo resolución a quienes jueguen el juego. Es importante además **no mostrar todo de una**, sino, ir introduciendo de a poco contenido, mecánicas, assets, etc.
- **Vale modificar el juego:** Se pueden construir features especialmente para la demo, que usualmente implican atajos o niveles acelerados.

Para inspirarse, pueden mirar demos de juegos en exposiciones en internet, como por ejemplo: https://youtu.be/T5Xx3MdqdgM.

Tener en cuenta que estas demos cuentan con recursos millonarios y varios meses de preparación, con lo cual son inspiraciones, pero no necesariamente se puede aspirar a construir espectáculos similares.

Exposición (40 minutos - duracionDeLaDemo)

La segunda parte de la entrega final consistirá en una presentación más tradicional. El contenido de la exposición será abierto, pero lo que se busca es transmitir la experiencia adquirida durante el desarrollo del BOSS, desde el lado del desarrollador.

A continuación se detalla una posible estructura de la exposición que no necesariamente se tiene que seguir estrictamente, pero que pueden tomar en caso de no encontrar una estructura mejor:

1. **Resumen de la idea del juego.** Contar en qué consiste el juego. Cuáles son los core aesthetics a los que se apeló. A quiénes podría gustarles el juego.
2. **Historia de la idea del juego**. Con qué idea se comenzó y cómo fue evolucionando. Cómo fue cambiando debido a problemas técnicos u observaciones mientras se jugaba. Cómo se fue completando.
3. **Elección de la tecnología.** Qué tecnologías se consideraron para construir el juego. Qué se observó en cada una. Qué prototipos se elaboraron y por qué se terminó eligiendo en la que se construyó.
4. **Organización del trabajo y toma de decisiones.** Cómo el grupo se organizó para la construcción del TP. Qué herramientas se usaron. Cómo se dividió el trabajo. Cómo se planificó qué construir en cada momento. Cómo se tomaron decisiones en relación a qué features deben incluirse en el juego, incluyendo decisiones de game design.
5. **Aspectos técnicos.** Qué partes resultaron fácil de implementar gracias a características técnicas. Qué partes resultaron difíciles. Cómo se sortearon las dificultades.
6. **Gestión de assets.** Cómo se obtuvieron y desarrollaron los assets. Qué assets funcionaron bien. Cuáles no tanto. Qué assets se tuvieron que adaptar. Qué herramientas se usaron.
7. **Exposición técnico-práctica (código)**. Mostrar la estructura del proyecto. Mostrar partes del proyecto que se consideren logros técnicos. Mostrar partes que deberían mejorarse.
8. **Conclusiones.** Qué se hizo bien durante el trabajo práctico. Que podría haberse hecho mejor. Qué cosas gustaron y se disfrutaron. Qué partes se padecieron. Cómo podría el equipo lograr un mejor trabajo. Experiencias más valiosas adquiridas.
9. **A futuro.** Qué features quedaron en el tintero. Cómo se continuaría el juego de tener más tiempo. Más allá de la materia, qué se puede extraer de lo construído.

La exposición será con proyector y puede acompañarse con imágenes, presentaciones de diapositivas, o cualquier medio que se considere adecuado. También puede usarse el pizarrón.