Universidad ORT Uruguay

Facultad de Ingeniería

Obligatorio I

Programación de Videojuegos I

Ingeniería en Sistemas

Programadores

Genaro Girardi - 199394

Karen Martínez - 168066

Diseñadores (Animación y Videojuegos)

Fabricio Danza - 190378

Martín Ivernizzi - 204373

Maximiliano Pereira - 184044

Docente

Joaquín Bello

Mayo 2018

Índice

1. Introducción

Para este obligatorio la consigna fue hacer un prototipo de juego de género plataforma, siendo programado en Haxe con FlashDevelop, en un plazo de 2 meses aproximadamente. Dicho trabajo se haría en conjunto con alumnos de la materia Producción 2, de la carrera Animación y Videojuegos.

2. Descripción del juego final

Para empezar, nuestro juego tendría las características de un supervivencia plataformero, de una única pantalla de 1920X1080, multijugador de 2 jugadores (en una misma máquina). El mismo estará disponible para browser (preferiblemente Firefox).

Objetivo/fin del juego
El objetivo del juego varía dependiendo de si somos el Jugador ó el Dios.
Si se es el Jugador, el objetivo será recolectar los objetos que se encuentran en el mapa (para esta entrega son monedas) para así poder acabar con el Dios.
Si se es el Dios, el objetivo será matar al Jugador antes de que este lo haga. Para esto dispondrá de diversas habilidades, siendo visibles en la parte superior derecha de la pantalla.

El juego termina cuando cualquiera de los dos jugadores muere, resultando en la victoria del otro.

Controles
El Jugador se moverá con las teclas ad, saltará con w, pudiendo efectuar doble salto, y disparará (en caso de ser posible) con la tecla ESPACIO. Al despegarse del suelo, el mismo se verá afectado por una fuerza en el eje y (simulando la gravedad), y eventualmente terminará en el piso nuevamente.
El Dios se moverá con las flechas (o con el teclado numeral activando la tecla NumLock), y seleccionará y usará las habilidades con el mouse. A diferencia del Jugador, el Dios no es afectado por la gravedad (ya que es un fantasma).

El menú principal se navega con las flechas, y se accede a cada opción con ENTER.

Con la tecla ESC se puede ir en cualquier momento al menú principal.

3. Procedimiento y justificación de diseño

3.1 Introducción

A continuación se dará una descripción sobre el procedimiento de la idea al diseño.

3.2 Construcción de la solución

En primera instancia, lo esencial era tener los dos personajes: el Dios y el Jugador. El tamaño de los mismos se vio definido por la resolución de pantalla indicada en el punto 2.

Una vez que se tenían los movimientos básicos de cada personaje, había que definir una forma en la que el Jugador pudiese ganar, y otra para el Dios.

Para el Dios se tenía pensado que tuviese habilidades. Las mismas debían incluir un proceso de “prueba”, en la cual se tiene que testear para cada habilidad su impacto en el juego (y el cómo interactúa con las anteriores). En caso de introducir una habilidad que pudiese llegar a “desequilibrar” el “poder” hacia un jugador, se tiene que pensar una forma de “equilibrarlo”; para así impedir un abuso de parte del Dios.

Condición de victoria/derrota: Primera instancia
Jugador: Se definieron los objetos que el Jugador tenía que recolectar (en primera instancia no había un criterio para colocarlos), y que al juntarlos pudiese disparar cierta cantidad de proyectiles. Si algún proyectil acertaba, ganaría el Jugador, en caso contrario volverían a aparecer las monedas (en posiciones aleatorias), y el Jugador debería volver a recolectarlas para tener la posibilidad de matar al Dios nuevamente. Los proyectiles del Jugador seguirían al Dios por un par de segundos, y luego mantendrían fija su dirección hasta irse del mapa.

Dios: Debido a que aún no se había pensado el cómo implementar las habilidades del Dios, lo que se hizo es que el Dios ganase si el Jugador colisionaba con él.

Condición de victoria/derrota: Segunda instancia
Jugador: Se pule la forma en la que se introducen los objetos recolectables al juego. Ahora el mapa tenía un tipo de tile que servía de indicador para colocarlos.

Dios: Se mantiene como estaba antes.

Una vez teniendo el escenario básico, se pasó a crear un menú principal simple. En sus inicios funcionaba con el ratón, luego pasó a funcionar únicamente con el teclado. Posteriormente se modifica la clase que permitía hacer funcionar los botones con el teclado para que pudiese soportar ambos (FlxButtonAnimation).

Condición de victoria/derrota: Tercera instancia

Jugador: Se pulen aspectos generales como la creación de objetos, y proyectiles del jugador. No se agrega más funcionalidad de victoria para el mismo a partir de aquí, por lo que no aparecerá la descripción de condición de victoria/derrota del Jugador en las siguientes instancias.

Dios: Se crea una clase que extiende de FlxButtonAnimation, llamada FlxButtonAnimationSkill para poder implementar la primera habilidad del Dios. Al principio toda la funcionalidad de la habilidad se encontraba en GameState.

La primera habilidad del Dios lanzaría un proyectil que iría desde el Dios hasta donde se ubicara el puntero del mouse al clickear.

Que el Jugador y el Dios colisionaran dejó de ser una condición de victoria para el Dios, y pasó a convertirse en un efecto de estado de inmovilidad para el Dios.
La segunda habilidad del Dios permitiría colocar una trampa para el jugador. Si el mismo la tocaba, pasaría a un estado de “inmovilizado”. La misma tendría ciertos criterios para ser colocada, tal como estar en una superficie, y estar a cierta distancia de los objetos recolectables, y del Jugador.

Condición de victoria/derrota: Cuarta instancia
Dios: Debido al gran acoplamiento que estaba teniendo GameState, se crea una nueva clase llama SkillsController, la cual pasaría a tener toda la manipulación de las habilidades del Dios. También se re ubica comportamiento que pertenecía al Dios que erróneamente estaba ubicado en GameState.

Los estados pasan a encapsularse en una clase.

Se refactorean nombres en forma general. Se define los atributos con el prefijo “v”, las constantes con “c”, y los parámetros de los métodos con “a”. También se sacan varios “números mágicos”.

Debido a ciertos inconvenientes que pudiese generar que la clase ButtonAnimationSkill no llamase al update de su padre, se optó por separar ambas clases.

Quedamos insatisfechos con la clase SkillsController, ya que la misma pasaría a tener demasiada responsabilidad en caso de seguir agregando habilidades; por lo que que se hizo otra clase llamada SkillLogic, la cual se extendería por cada nueva habilidad, englobando la lógica de cada habilidad en una clase a parte. Esto permitiría a SkillsController manipular las habilidades sin saber lo que hace cada una, funcionando únicamente como un manipulador/administrador de habilidades.

Se piensa que podrían haber cosas a mejorar en cuanto a performance; por lo que se seguirá trabajando en ello luego de esta entrega.

También se detecta que el sonido importado a veces no se escucha; por lo que intentaremos solucionar esto también.

Descripción de las habilidades del Dios implementadas
A continuación se otorgará una breve explicación sobre cada habilidad implementada (del Dios). Las habilidades no tienen nombre en el juego, pero para la documentación tendran para poder diferenciar una de otra..

Proyectil: Esta habilidad dispara un proyectil que va desde el Dios hacia donde se encuentra el puntero del ratón. El mismo continúa hasta colisionar con el Jugador o se vaya del mapa. Si colisiona con el Jugador; el Dios habrá ganado la partida.

Trampa: La misma consta de una trampa que sólo se puede colocar adherida a una superficie y a cierto radio del Jugador y los objetos existentes en el mapa. La misma se teñirá de rojo si no se puede colocar, y verde en caso contrario. Si el jugador llega a tocar alguna trampa, el mismo pasará a estar inmovilizado un par de segundos.

Bomba: Se lanza una bomba que cae desde el dios hasta golpear una superficie, o al Jugador. Para poder lanzar la bomba se requiere que el Dios esté de color verde porque la misma no se puede activar si el Dios se encuentra demasiado cerca del Jugador.
Si el Jugador colisiona con la misma; el Dios habrá ganado la partida.

4. Requerimientos/Alcance

4.1 Introducción

A continuación se dará una breve explicación de los requerimientos pensados para el juego final. Los mismos son a modo general (siendo los primeros los más prioritarios), ya que la forma en la que se plasman corresponden a más bien al punto 2.2.

4.2 Requerimientos

Consideramos que el juego final llegaría a ser satisfactorio si cumple con los siguientes requerimientos:

R1: Tener un Jugador funcional (movimiento).
R2: Tener un Dios funcional (movimiento).
R3: Tener al menos una condición de victoria tanto para el Dios como para el Jugador.
R4: Hacer al menos 3 habilidades funcionales para el Dios.
R5: Tener un menú principal.
R6: Tener una forma de salir del juego sin necesidad de cerrarlo.
R7: Que el juego posea música de fondo, y efectos de sonido en los personajes.

En caso de cumplir con todos antes de tiempo, pasaremos a pulir la perfomance, calidad de código, y agregar requerimientos que sumen a la calidad del trabajo (pero que no serían los esenciales).

Como requerimientos adicionales:

RA1: Tener varios escenarios para darle variedad a los jugadores.
RA2: Que el Dios antes de jugar pueda optar entre varias habilidades, las cuales pasarían a ser las que usaría en el juego.

4.3 Alcance del prototipo actual

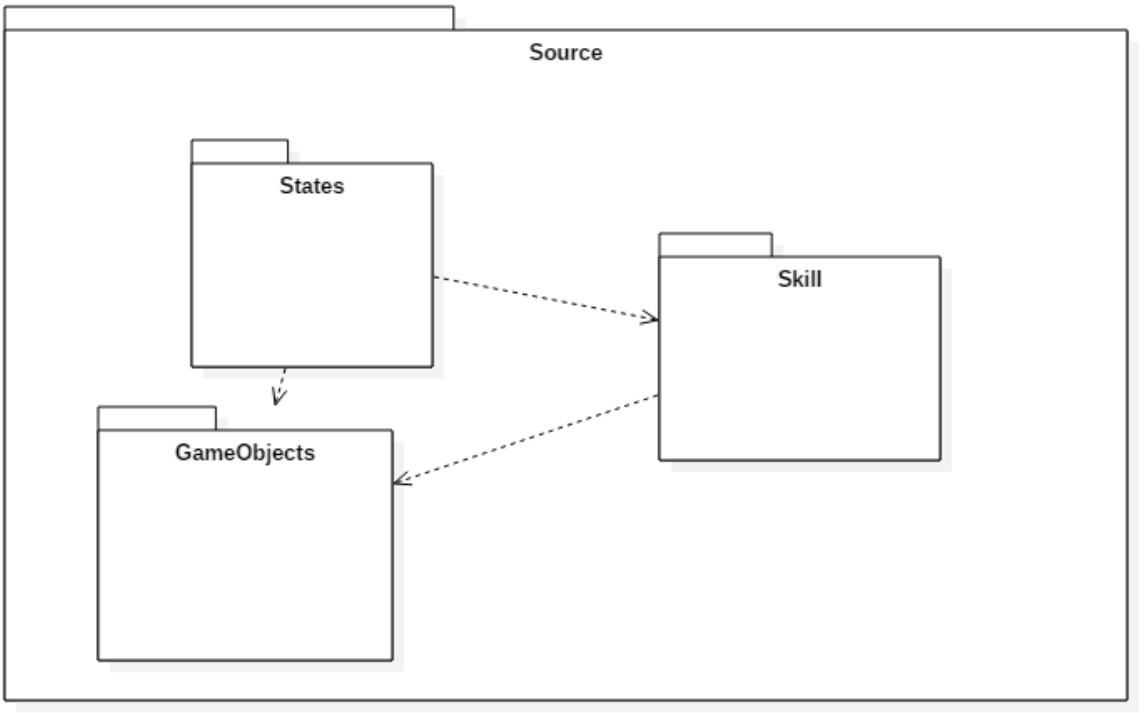
Nuestro prototipo cumple con todos los requerimientos más importantes(R1-R6 y parcialmente R7). Esto no quita que los mismos no puedan ser pulidos para la 2nda entrega.

5. Diagramas

A continuación se presentará un diagrama básico de Clases/Paquetes de la solución. Se omitirá lo que no sea pertinente, ya que el objetivo es ver la relación entre las clases.

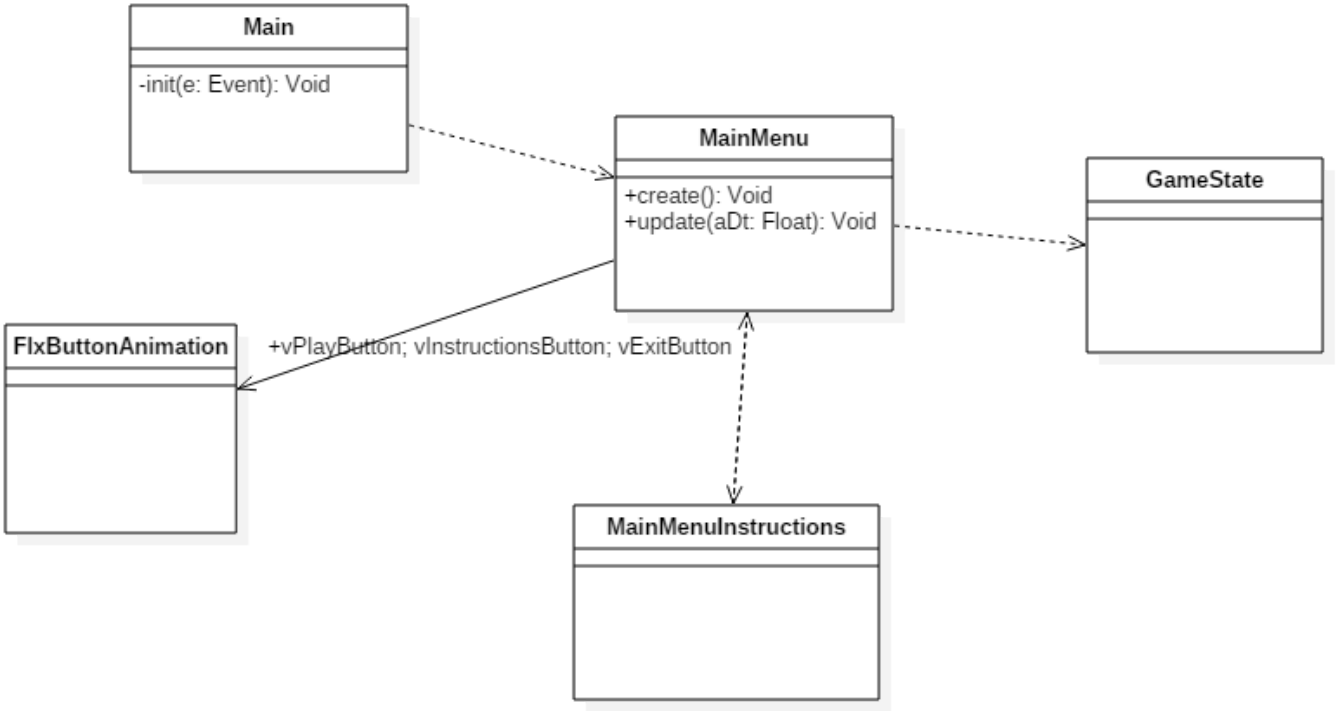
5.1 Diagramas de Clase/Paquetes

5.1.1 Diagrama de Paquetes general

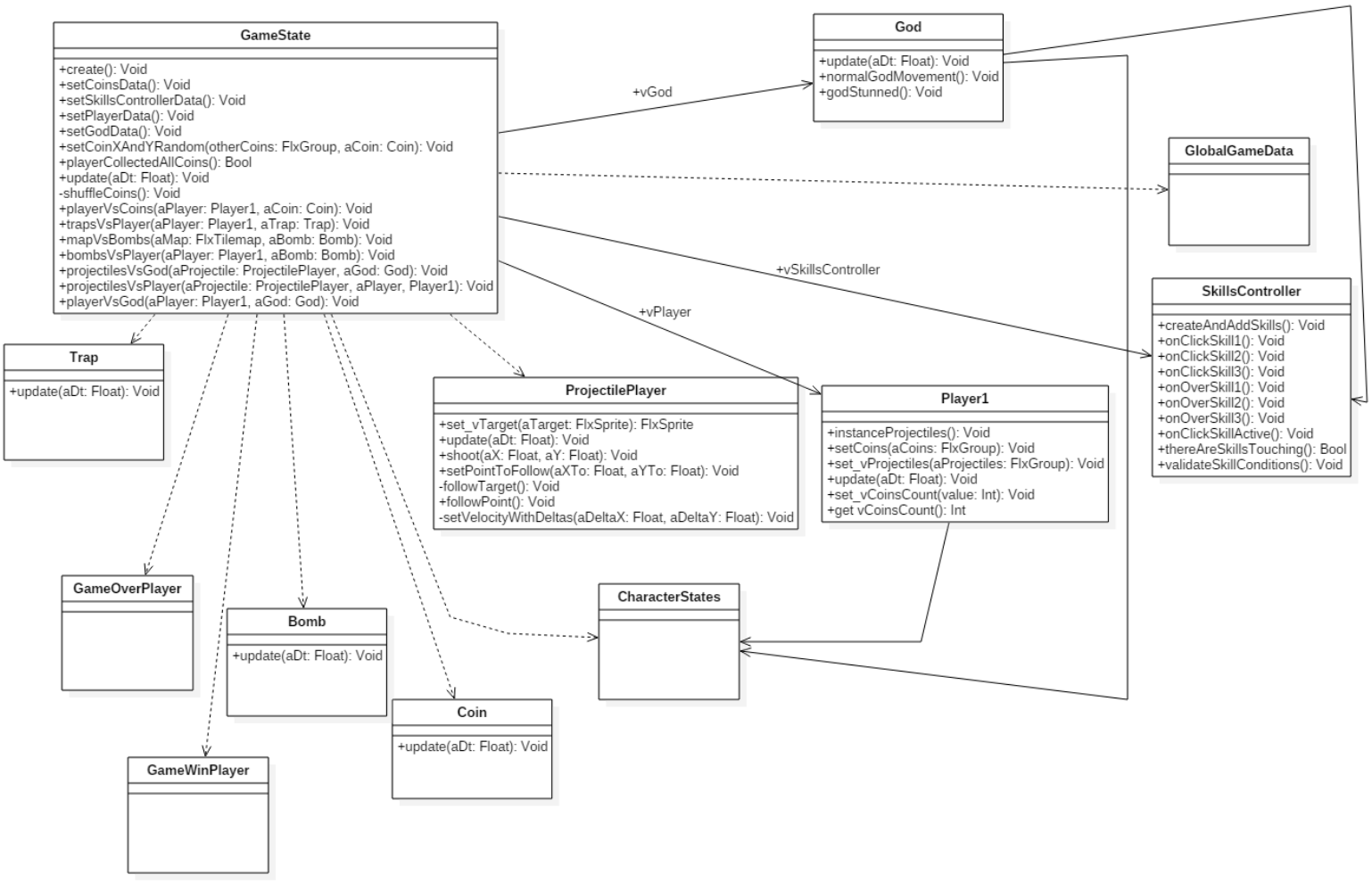


5.1.2 Diagrama de Clases

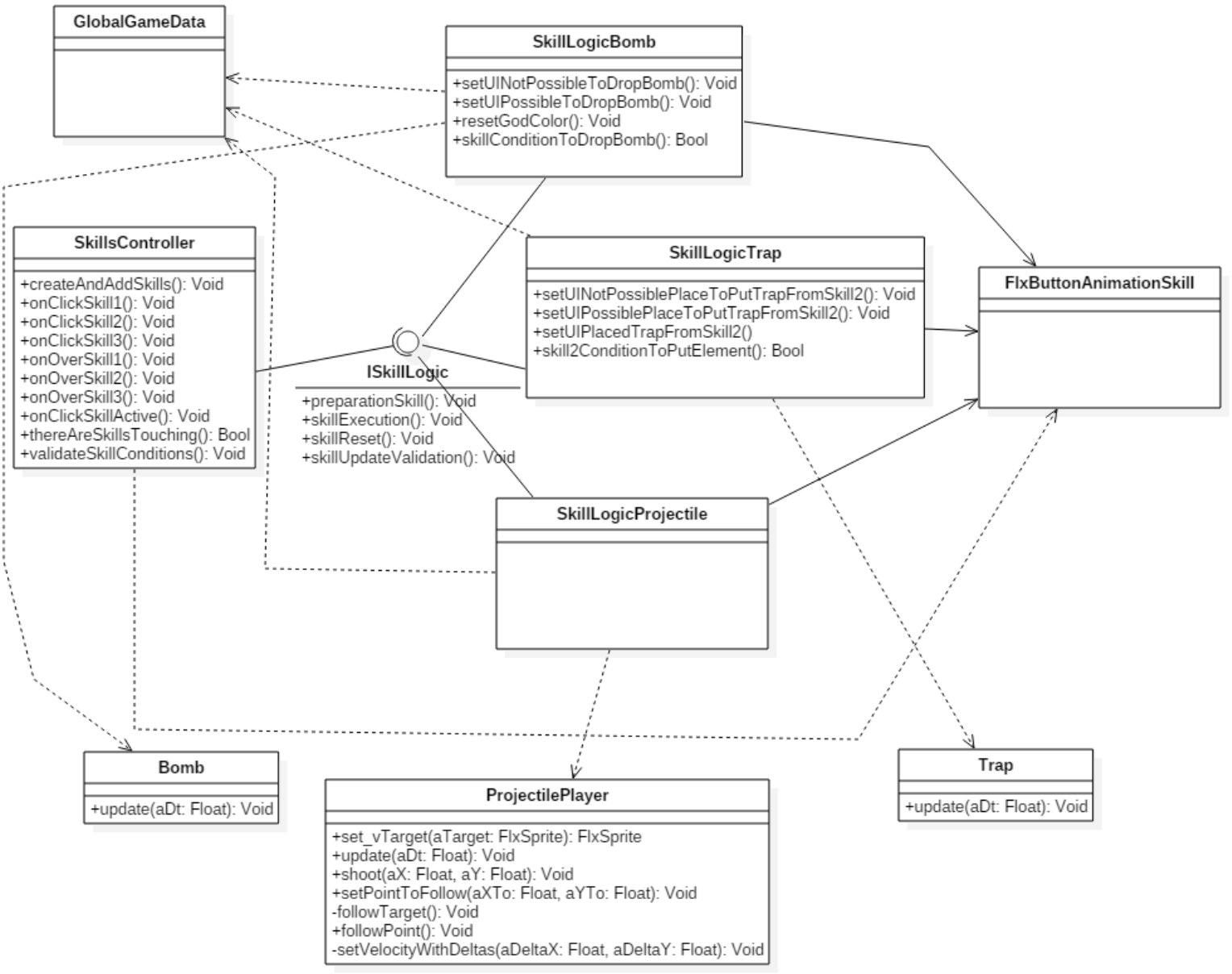
Diseño de las clases necesarias para correr el menú principal:



La clase GameState es la que se encarga del manejo del estado del juego, utilizando varias otras clases. Este es un diagrama de la interacción del GameState con las clases que representan los varios elementos del juego:



Para la implementación de las habilidades, la clase SkillsController no se responsabiliza de saber cómo funcionan las habilidades que usa, sino que llama a la lógica de la habilidad adecuada cuando el GameState le solicita las Skills. Esto lo hicimos para facilitar la extensibilidad del sistema de habilidades del dios en un futuro.



6. Repositorio

El repositorio del proyecto (al igual que esta documentación), y el proceso de la construcción del mismo se puede encontrar en: https://github.com/XxKarikyXx/ProgVJ1