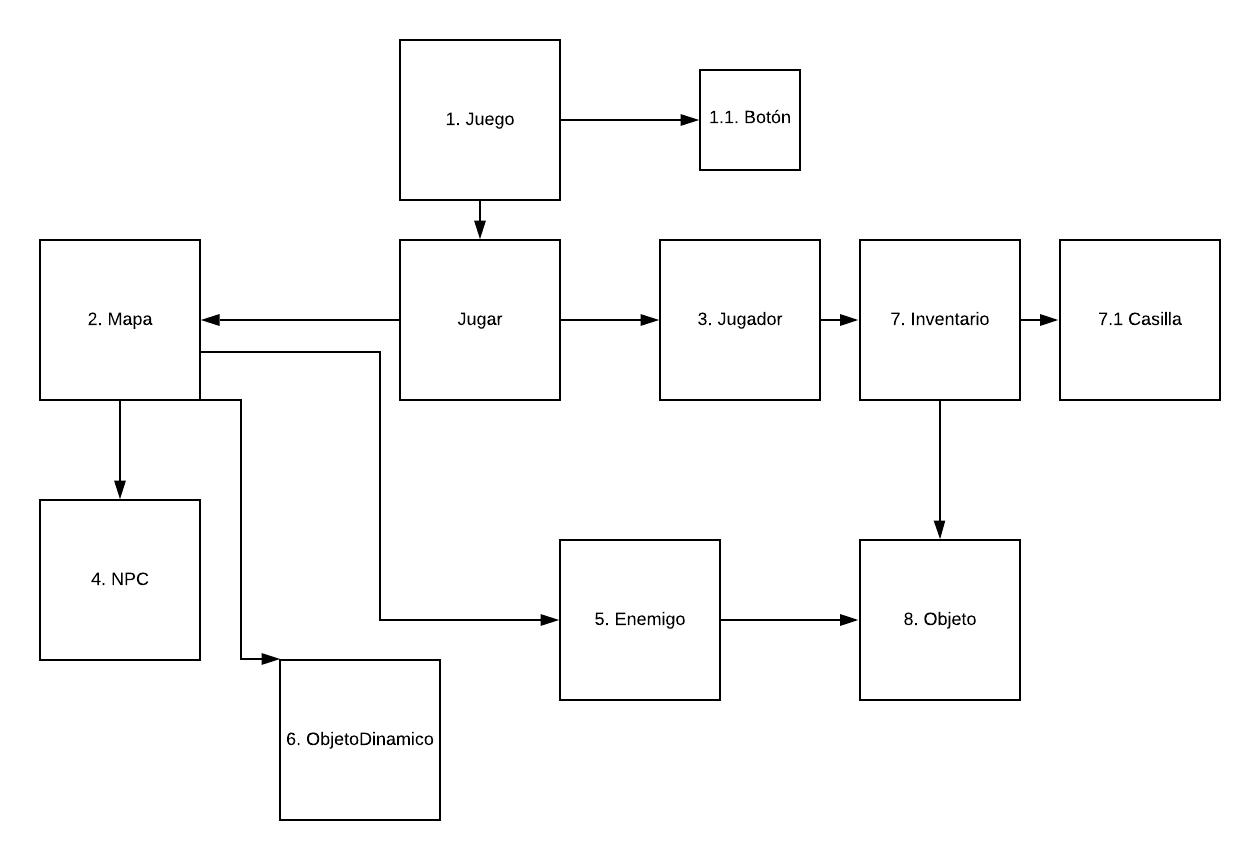
Diagrama de Clases

Descripción de las clases:

1. **Juego:**

Hereda la clase *QGraphicsView*

Métodos importantes:

1. Menú Pantalla Inicial.
2. Menú Pausa.
3. Inicializar el juego.
4. Cerrar Menú Pausa.

Atributos relevantes:  
1. Bool Juego Activo.  
2. Pantalla  
3. Enemigos Actuales.  
4. Objetos Suelo  
5. Mapas  
6. Mapa Actual

La clase juego será la que inicialice y de las diferentes opciones para el jugador, de las cuales las más importantes serían permitirle jugar una nueva partida o cargar una previa. También incluirá los métodos que permitan la pausa del juego en curso.

Contendrá una lista con todos los mapas disponibles para el jugador, una con los enemigos actuales y una última con los objetos en el suelo.

* 1. **Botón:**Clase auxiliar.Hereda las clases *QObject* y *QGraphicsRectItem*  
     Cuando se de click en un objeto de esta clase se emite una señal que será usada desencadenante.

1. **Mapa:**Hereda la clase *QBrush*

Atributos Relevantes:

1. Fondo del Mapa
2. Paredes del mapa
3. Objetos dinámicos del mapa

Esta clase construye el mapa que recorrerá el jugador. Tendrá 2 imágenes: Una de fondo y otra de “pared”   
Contendrá una lista de pares ObjetoDinamico-Coordenada que se usará para organizar dichos objetos.

1. **Jugador:**

Hereda las clases *QObject* y *QGraphicsPixmapItem*Métodos importantes:

1. Cargar Sprites
2. Pausar
3. Despausar
4. Herir
5. Atacar
6. Agregar Objeto al inventario
7. Actualizar físicas
8. Mover
9. *keyPressEvent*

Atributos relevantes:

1. Sprites
2. Estados (Sí/No se mueve, Sí/No está atacando, etc)
3. Atributos físicos
4. Dirección
5. Estadísticas (Ataque, Vida, Defensa)
6. Inventario
7. Timers
8. “Tiers” de estadísticas

Esta clase tendrá la jugabilidad e interacciones con los demás objetos. También hace un par de *enum* para hacer un poco más fácil el manejo de los sprites y las diferencias entre el primer y segundo jugador.

Los tiers serán ints que determinarán los rangos de estadísticas de los enemigos

1. **NPC:**

Hereda la clase *QGraphicsPixmapItem*

Métodos importantes:

1. Interacción con Jugador
2. Cargar Dialogos

Atributos relevantes:

1. Texto en pantalla (Bool)
2. Texto a decir

Esta clase recibirá un ID para diferenciar los diferentes NPC’s y sus interacciones con él

1. **Enemigo:**

Hereda la clase *QGraphicsPixmapItem*

Métodos importantes:

1. Herir
2. Pausar
3. Despausar
4. Mover
5. Animar
6. Actualizar Fisicas

Atributos Relevantes:

1. Estadisticas
2. Objeto de “drop”
3. Timers
4. Atributos Fisicos
5. Sprites

Esta clase recibe un int como ID para cargar su tipo de enemigo y sus sprites. En el momento de creación sus características son determinadas en rango al tier del jugador.

1. **Objeto Dinamico:**

Hereda la clase QGraphicsPixmapItem

Métodos importantes:

1. Actualizar Fisicas
2. Dar aceleración
3. Colisiones

Atributos Relevantes:

1. Atributos físicos
2. ID

Esta clase recibe un int como ID para determinar la imagen y atributos que tendrá

1. **Inventario**

Métodos importantes:

1. Agregar Objeto
2. Mostrar Inventario
3. Cerrar Inventario
4. Dibujar Inventario

Atributos relevantes:

1. Inventario Abierto (Bool)
2. Mapa de objetos
3. Lista de casillas
4. Lista de *QGraphicsTexItem* con números
5. Cantidad máxima de montones de objetos

Esta clase dará las funciones para revisar el inventario del jugador

* 1. **Casillas:**Clase auxiliar para la clase Inventario

Hereda las clases *QObject* y *QGraphicsRectItem*

Esta clase pondrá una imagen dependiendo del int ID que se le pase

1. **Objeto:**

Hereda las clases *QObject* y *QGraphicsPixmapItem*

Métodos importantes:

1. Mostrar en mapa
2. Recoger

Atributos relevantes:

1. Imagen
2. Posiciones
3. Cantidad
4. ID
5. Timer

Esta clase cargará características diferentes dependiendo del int ID dado.

|  |  |
| --- | --- |
| Cronograma | |
| Primera Semana:  Lograr la interacción entre los jugadores y el medio de manera efectiva. | |
| 7 de Julio:  Lograr que los jugadores puedan atacar a los enemigos de manera orgánica. Y que estos al ser derrotados tengan probabilidades acordes para soltar los objetos de progresión del juego. (1 Día) | 8 de Julio:  Lograr la interacción del jugador con los diferentes NPC’s del mapa principal. Se buscará también una cantidad adecuada de recursos para la progresión de dificultad del juego. (1 Día) |
| 9 de Julio  Lograr una correcta interacción entre los diferentes objetos repartidos en el mapa con el jugador. En este apartado haciendo énfasis en las físicas. (2 Días) | 11 de Julio  Por último, organizar el desplazamiento entre los diferentes mapas del juego y sus respectivos bloqueos por progresión de juego. (2 Días) |
| Segunda Semana: Optimizar a los enemigos y los objetos del mapa | |
| 13 de Julio  Agregar variedad de enemigos con sus respectivos objetos de mejora. También se intentará hacer que los diferentes enemigos tengan efecto sobre la jugabilidad en el momento de ser derrotados. (2 Días) | 15 de Julio:  Agregar IA’s para los diferentes enemigos agregados.  Organizar los recursos audiovisuales de cada enemigo.  (3 Días). |
| 18 de Julio  Agregar de manera coherentes obstáculos en los diferentes mapas. (2 Días) | |
| Tercera Semana:  Optimizar el código y corregir posibles errores | |
| 20 de Julio  Se analizará el código en el orden previo en busca de los diferentes apartados que se puedan mejorar o corregir. Corregir todos los posibles errores que puedan ser encontrados.  Optimizar diferentes aspectos de jugabilidad, tanto mapas, enemigos y recursos usados (7 Días) | |
| 27 de Julio  Finalización del proyecto. | |