Juego basado en puzles tipo portal, pero en 2D. El jugador tendrá que ingeniárselas para superar las pruebas a propuestas en cada mapa y lograr activar los “checks” de nivel validado. El jugador se moverá a través del mapa para hallar la manera de avanzar al siguiente nivel. También se podrá completar el nivel de al menos 2 maneras diferentes.

Será un juego en 2D con enfoque en las físicas y en las interacciones entre los objetos desarrollados. El plan inicial es lograr que haya estos objetos:

Jugador: Que se pueda mover a través de impulsos de aceleración, tanto de los inputs del que lo controla, como por impulsos de otros objetos.  
Arma de gravitación: Que repela y atraiga objetos dinámicos. Inicialmente se podría plantear que, al sostener un clic, se cree una zona de aceleración (invisible) lo suficientemente grande como para atraer al objeto sin necesidad de implementar mucho más que la zona. Que repela se podría lograr con una función que lea el clic y aplique una aceleración lo suficientemente grande para mover al objeto, aunque tal vez sea más adecuado aplicarle una velocidad.

Zonas de aceleración: Zonas que apliquen una aceleración constante a los objetos dinámicos que se encuentren en su zona de efecto. Se implemente una opacidad para que el jugador vea la dirección de la zona.

Objetos Dinámicos (Cajas que se puedan mover con el arma, por ejemplo)

Objetos Estáticos: A pesar de su nombre, no se plantea que se queden necesariamente en un lugar del mapa siempre, la parte de estático iría más bien ligada al hecho de que no se verían afectados por zonas de aceleración.

Puntos de Gravedad: Se podría ver como una zona de aceleración radial, que emule atracción o repulsión. Estos puntos no deberían chocar directamente con los objetos dinámicos, sólo darles aceleraciones.

Segunda iteración:

Se ahondará un poco más sobre las clases usadas y las ideas generales para cada una:

ObjetoEstatico:

Heredará QGraphicsPixmapItem y QObject.

Se usará como clase base para objetos que no serán afectados directamente por la gravedades o aceleraciones de algún tipo. Se espera que tengan colisiones con otros objetos; se podrán escalar para, por ejemplo, hacer paredes, plataformas, etc; tendrá un tamaño y un ángulo de rotación. Guardará también la imagen que se usa en el objeto.

Plataforma:

Hereda ObjetoEstatico.

Su constructor es una copia de ObjetoEstatico, e incluye una variable “colorPlataforma” que se usará para identificar el tipo de plataforma.

ObjetoMovible:

Heredará QGraphicsPixmapITem y QObject.

Estos objetos podrán ser movibles a través de aplicaciones de velocidades y/o aceleraciones.

Su constructor incluye que tipo de objeto será generado (jugadores, cajas, cubos, esferas, etc), una masa, la posición, la velocidad inicial y la aceleración inicial.

Más allá de guardar las variables usadas para calcular los movimientos, también incluirá un QRectF para saber su caja de colisiones, sus sprites, el frame actual, etc.

ZonaGravitacional:

Heradará de QGraphicsPixmapItem y QObject.

Se espera guardar la información relevante para calcular valores de aceleración aplicados a objetos movibles. Inicialmente se tendría guardado FuerzaGravitacional y la rotación de la zona.

Se calcularía entonces con las funciones trigonométricas las aceleraciones aplicadas en x y y para los objetos dentro de los límites de la zona.

Arma:

Heredará de QGraphicsPixmapItem y QObject

Tendrá 2 tipos de disparos inicialmente, atractivos y repulsivos. En la implementación de los disparos inicialmente se podría usar para el disparo atractivo una ZonaGravitacional que exista mientras se da un clic, dicha zona tendría una fuerza grande e iría desde la posición del arma hasta el primer objetoMovible encontrado. Para el repulsivo, se puede implementar que, mientras el objetoMovible esté pegado al arma, se pueda aplicar una aceleración grande.

Jugador:

Hereda de ObjetoMovible.

Su constructor es una copia de ObjetoMovible, pero en este se agrega un objeto extra Arma.

También necesita sobreescribir una función virtual heredada de ObjetoMovible (setPos) para que cuando se le aplique también mueva la posición del arma.

Mapa:

Tendrá listas de instancias de los objetos descritos usados para cada mapa en su construcción.

Juego:

Heredará de QGraphicsView

Tendrá Timers, las instancias para la pantalla, para el jugador, listas de instancias de mapas. Funciones para la interacción de diferentes tipos de objetos, ejemplo: void InteraccionZonas(Zona \*, Objeto \*) que se encargaría de aplicarle una aceleración al objeto, entre otras.

También sobreescribiría funciones virtuales para poder manejar la interacción del jugador con los elementos interactivos del juego. Tales como mousePressEvent(), mouseMoveEvent(), keyPressEvent(), etc.