TITLE PAGE

Class Structure

I chose to use the singleton design pattern for various parts of the game. This allows me the ability to…

WHAT ABOUT ABC FOR SINGLETON? Find out more and see what I should do. Probably unnecessarily complex.

InputController

Allows debounce by trapping the state of the player’s fire key every frame, to ensure that they aren’t able to fire repeatedly by holding down the button.

IRenderable

An interface designed to make rendering simpler, IRenderable is derived for all objects in the game which are drawn.

Player

Contains the information that pertains specifically to the player object i.e. health.

Debug

This class was created in order to more easily facilitate the drawing of debug information to the screen. As I understand it, text is drawn with IDiceInvaders::drawText , and when another string is drawn over the top of this, the original will be culled. I don’t directly have the ability to remove text from the screen, but I will need to update strings in position in the array. As such an STL map seemed to be the best option for storage of these strings. This allows me to very simply push a new string to the debug renderer, and also to access specific members of the map by index.

Player

The player is only able to fire a single rocket at once in the original game, and I have replicated that for use here.

EnemyManager

This class was created to make the process of spawning, updating, and managing the waves of enemies easier. In order to spawn the familiar shape of enemies, we use a 1D std::list. This is used over a 2D container as memory management is far easier this way, and it affords us a much more simple interface to work with.

I am using the “enemy2” sprite for the front rows, as it is 2 pixels wider than the other sprite. This means that the player is correctly rewarded a greater number of points for hitting the smaller enemies.

Collision

Not using groups/masks. Would normally. Instead here hardcoding it for simplicity (i.e. only collide player with bomb/enemy, rocket with enemy).

Using (hardcoded) clipping rectangles for collision to ensure that we don’t get false positives for intersections based on the sprites.

**Is it faster to use circle collision first?  
Mention that I could use collision groups/masks  
Mention the fact that I would use dist^2 over distance to save on sqrt operation**

**Issues so far**

Memory leaking from Player destructor. And the same fucking thing in PlayerUI.

Leak possible within ProjectileManager.

Memory leaking from EnemyManager. Every single time I spawn an enemy, I am leaking its memory. I can’t work out why. So that’s fun.