Game Design Document

Bombs Away is a simple game with a premise that is fun and easy to learn. The main goal of the game is to get to the key for each level and then get the key to the finish area. The main obstacle are the crates. To get past the crates, the player has to throw bombs at them to blow them up, but watch out, the boxes throw splinters that cause damage to the player. Take to much damage and you have to restart the level.

This game was made using the Godot game engine. All code and art in the game was created by Jared Harmon besides the music, which was found at Fesliyan Studios website, which claims that it is royalty free.

PLAYER

The main scene of the game is the player scene. The player is always in one state. The player takes in input for movement and left mouse clicks. The arrow keys and mouse click are read every frame and movement is then calculated from there. If the player clicks, a bomb is generated and starts to move towards the mouse location. The sprite also changes depending on what direction the player is moving. Nodes attached to the player scene are as follows.

Sprite – Used to represent the player visually on the screen
Camera2D – Locked to the player
CollisionShape2d – Used to detect collisions with walls and splinters
HealthBar – Used to visually represent the amount of health left
RichTextLabel – Used to display how much longer is left on the level before restart

BOMB

This scene contains all info and logic pertaining to the bombs. The bombs are instanced from the player scene. When they are instanced, a start function is called which instantiates all the need info most importantly, the vector for the bomb to follow. After 1 second the Explode function is called. This function calls the take damage function on all crates inside of its range and on the player if they are in range. It then removes itself from the scene tree.
Nodes attached to the bomb are as follows

Timer – for counting down till the bomb explodes
AnimatedSprite – For playing the ticking animation
AudioStreamPlayer – Not working, supposed to be a bomb sound
CollisionShape2d – For detecting collisions with boxes.

CRATE

Crates present the main obstacle for the player. When hit, the crates generate 20 splinters which fly off in random directions. It passes itself to a variable, deletes itself, then generates 20 instances of splinters.
Nodes attached to crate

Sprite
CollisionShape2d

SPLINTER

This class is the main damage dealing class and the primary threat to the player. When the instance is created the start function is called which sets the position and passes a reference to the player. The velocity is randomized and the splinter is sent off at a random direction. This is achieved by generating a random vector with x and y coordinates between -10 and 10. The vector is then normalized and multiplied by the speed. After 5 seconds or 10 bounces, the splinters are then removed from the tree.

KEY

This class is simple and just consists of a sprite and some basic code to change the state of the players has_key variable. When the player enters the area2d, the node is removed from the tree.

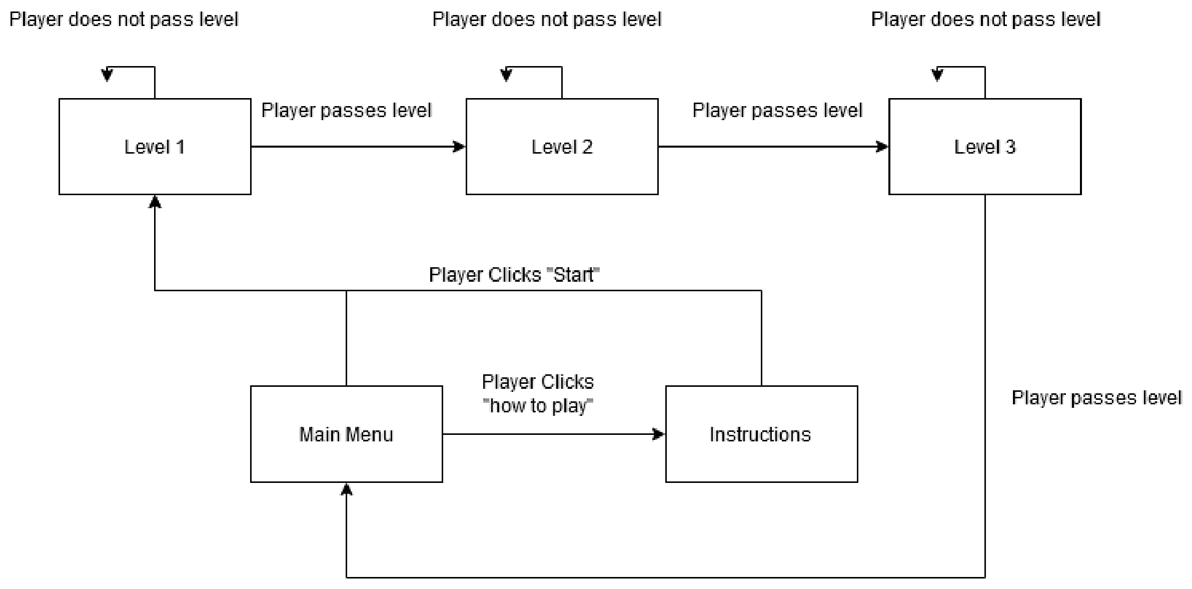
FINISH

This node is simple and just checks that the player is in the area2d and if he has the key, if both are true it advances to the next level.

STATE TRANSITION DIAGRAM

Individual components were not designed with the idea of state transistor, but the player and the direction they are moving and whether they have the key or not could be represented as a state transition diagram.

Here is a state transition diagram for how the levels are changed in and out based on the player actions.



Software Engineering Plan

As this is a solo project, I did everything in the game by myself. Everything was made by me except for the base engine and the music. I made all the sprites using the Pyxel software which is a nice software package.

My original plan for the final project was to make a spaceship fighting game using “real” orbital mechanics. I have trouble doing it in Godot due to my lack of understanding of the tree structure that the engine uses. My plan was as follows

Week 1: Exploratory work. Decide what I wanted in my game.
Week 2: Base Mechanics and Game play
Week 3: Final Touches, Music, Sprites, Polish

That plan didn’t happen.

I spent the first week piddling with my first game idea, I had a lot of trouble with it and decided to abandon it. I then messed around with tile maps. After I got tile maps working, I realized I spent very little time on the actual game, so I started messing around with stuff bouncing off walls and that is how I got my idea for the game. I think it turned out well in the end.

The actual work was compressed and the exploratory phase lasted a lot longed then planned.