GAD173

Vishnu Raveendran

5000517

Technical Design Documentation

First Build

Contents:

1.1 Overview

1.1.1 Purpose

1.1.2 Scope

1.1.3 Deliverables

1.1.4 Tools, target platform and system requirements

1.2 Program

1.2.1 User Interface

1.2.2 Functions and program layout

1.2.3 Loading from files

1.2.4 UML Class diagram

1.2.5 UML Case diagram

1.2.6 Timeline

1.3 Future features

Purpose:

To make a breakout clone and make a map editor for the game.

Scope:

The map editor will allow the user to create their own breakout level. It will allow them to drag tiles into place or select tiles and save the level into a file.

Maps will be saved into a text file which can be loaded from the game.

The game will be a clone of the game “Breakout”.

Deliverables:

These are the deliverables for the first build of the editor and the game:

- 32 x 32 grid
- User must be able to place bricks wherever they see fit
- User must be able to save the map to a file
- User friendly UI
- The game must load map from the text file
- Game must have core mechanics done

Tools:

- Visual Studio (Using C++, SFML libraries and kage2d engine)
- Source tree
- GitHub
- Target platform: Windows

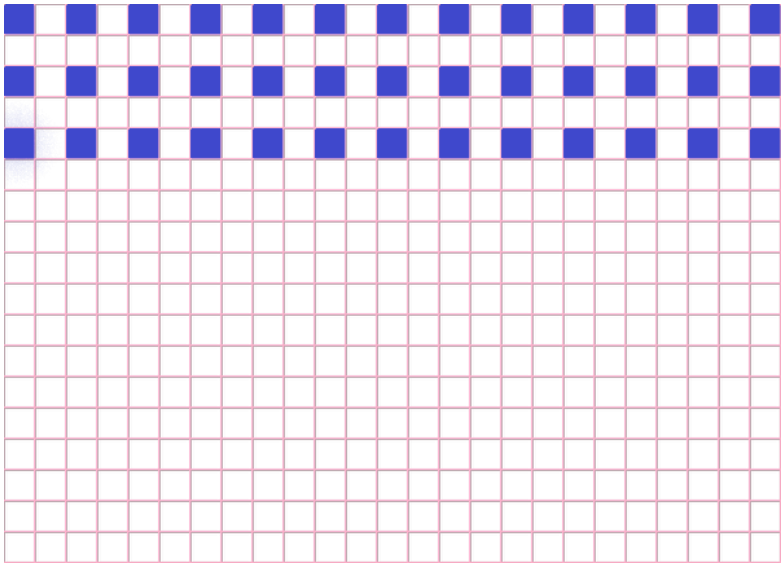
Requirements:

- OS: Windows 7 and above
- Processor: 2 GHZ
- RAM: 4 GB
- Graphics: 256MB
- DirectX: Version 10
- Storage: 50MB of free space

UI:

There will be a dropdown menu with a save option, load option, reset option and exit option.

Blocks will be draggable onto a 32x32 grid or there will be a 32x32 grid with selectable squares (where marked squares represent blocks and unmarked will be left blank).



Functions:

The game will focus on the main function and the update function. Where update will handle things such as collision of the ball and the brick and the deletion of blocks and score while main will initialize all starting sprites and values and the main menu.

The main menu will also have the map editor tool in it thus handling all functions related to the map editor.

Some functions are Save(), Load(), Reset(), StartGame(), and so on.

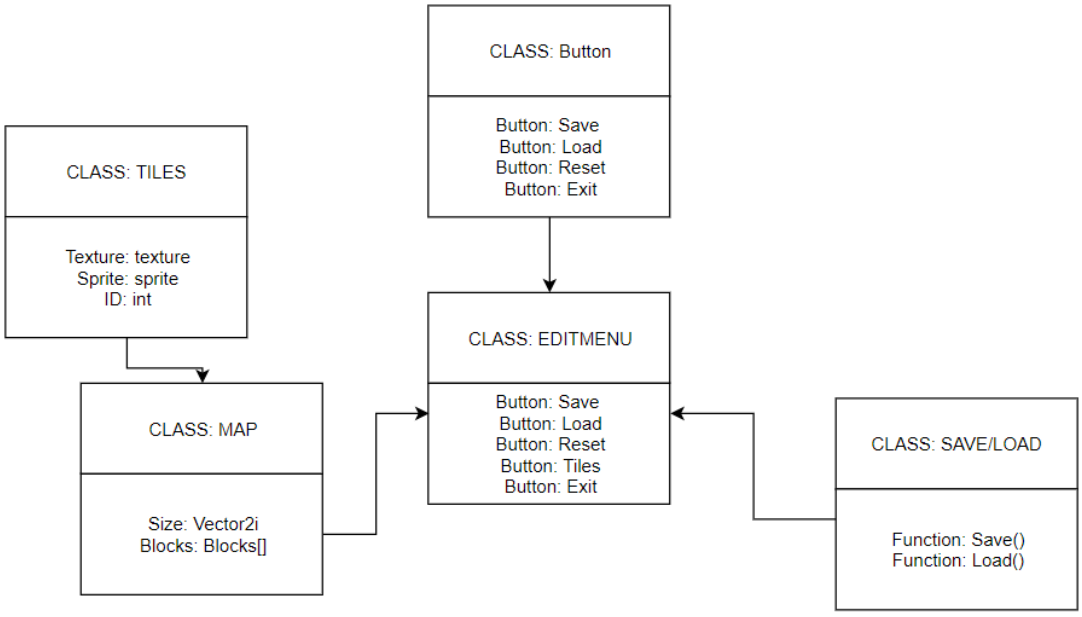
Loading from files:

Maps will be loaded from a text file. It will be read in array form.

```
1  [Map]
2  0, 1, 0, 1, 0, 1, 0, 1, 0, 1
3  1, 0, 1, 0, 1, 0, 1, 0, 1, 0
4  0, 1, 0, 1, 0, 1, 0, 1, 0, 1
5  1, 0, 1, 0, 1, 0, 1, 0, 1, 0
6  0, 1, 0, 1, 0, 1, 0, 1, 0, 1
7  1, 0, 1, 0, 1, 0, 1, 0, 1, 0
8  0, 1, 0, 1, 0, 1, 0, 1, 0, 1
9  1, 0, 1, 0, 1, 0, 1, 0, 1, 0
10
11 [Ball Spawn]
12
13 x: 50 y: 30
14
15 [Paddle Spawn]
16
17 x: 50 y: 10
```

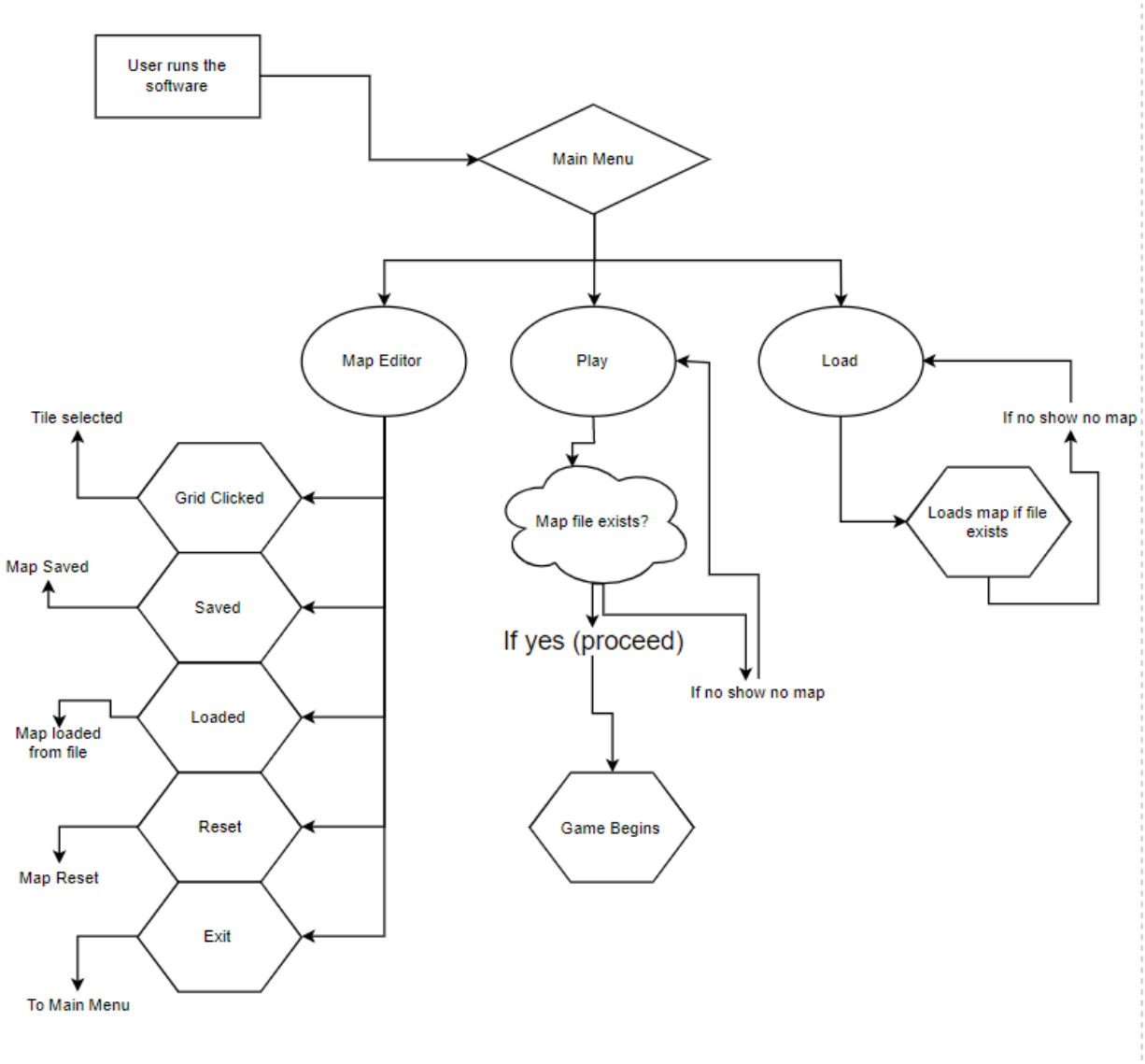
UML Class:

UML Class diagram for the editor

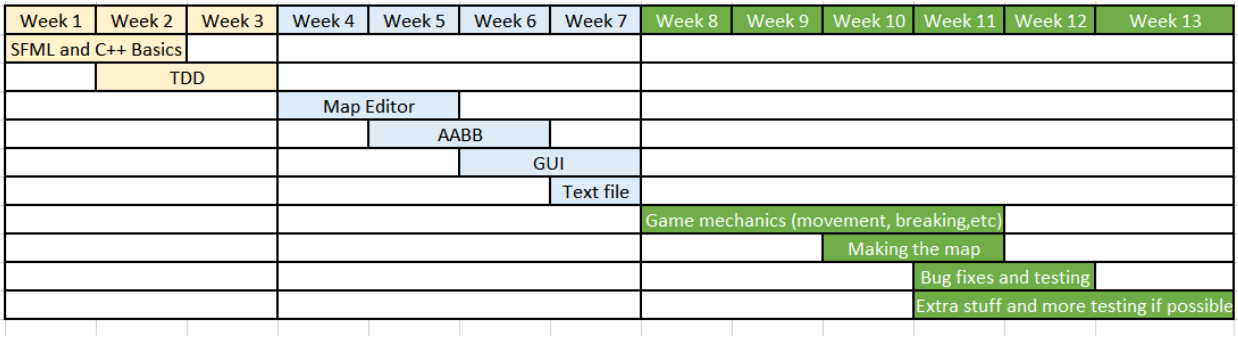


UML for the game will be added in a later build

UML Case:


User runs the software
Main Menu
Map Editor
Play
Load
Tile selected
Grid Clicked
Map Saved
Saved
Loaded
Map loaded from file
Reset
Map Reset
Exit
To Main Menu
Map file exists?
If yes (proceed)
If no show no map
Game Begins
Loads map if file exists
If no show no map

Timeline:



Future Features (if possible):

- Multiple files
- Using a 2D array
- Powerups
- More GUI functions