CS360 – COURSE PROJECT

Members: Pham Tran Tri – 1575301

Doan Dinh Cuong – 1575294

Ngo Truong Minh – 1584632

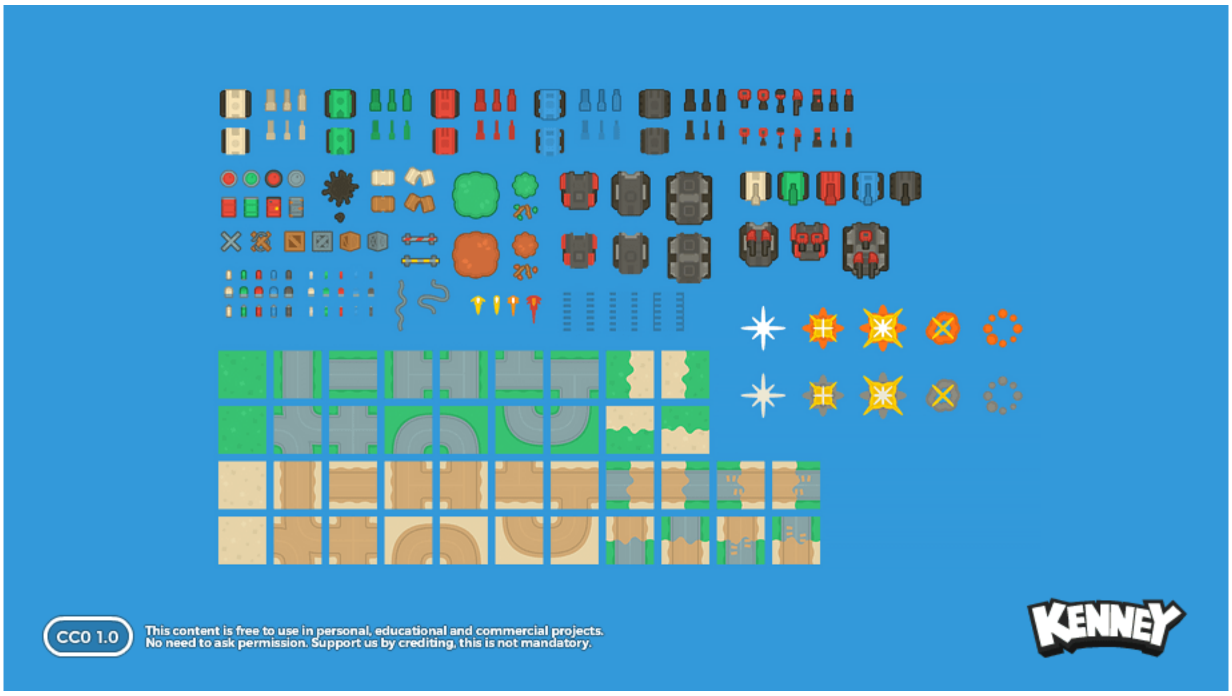
Dao Trung Hieu – 1575372

Nguyen Trung Kien – 1575392

2D Tank

I. Introduction and Overview

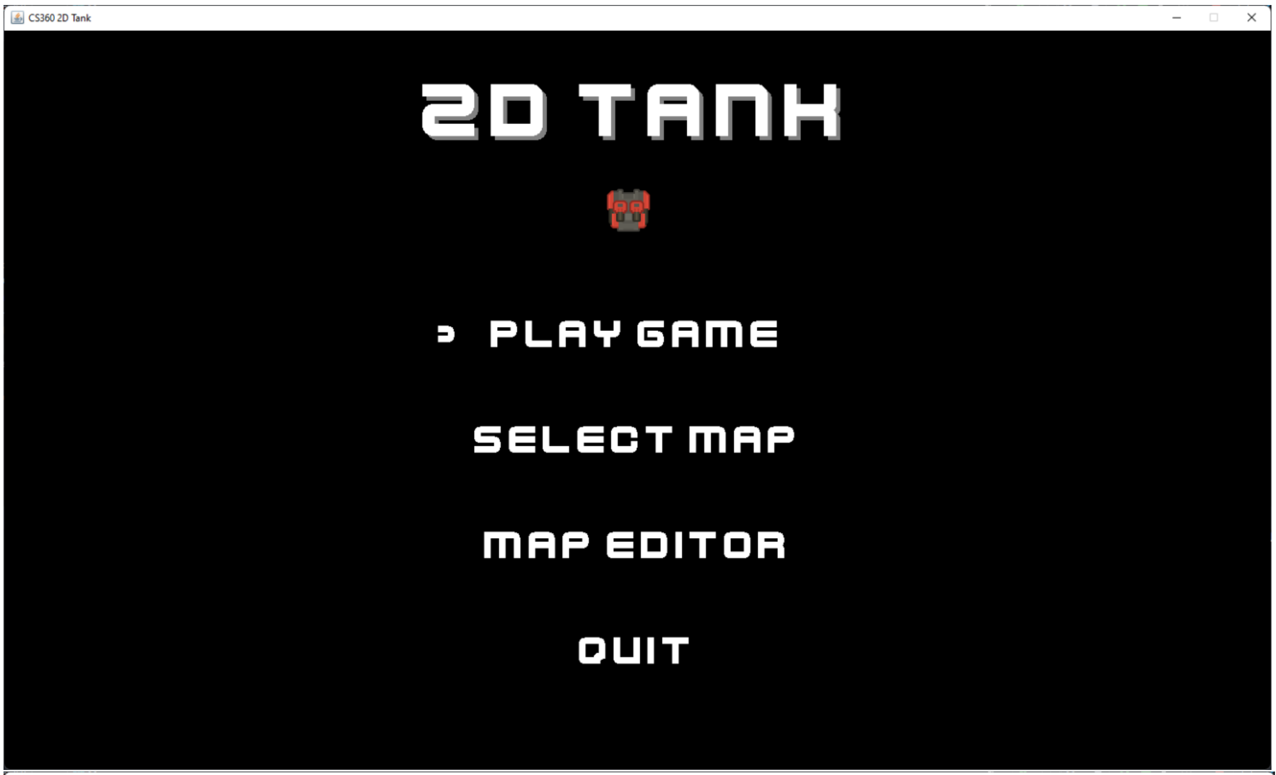
Based on the concept of the classic game The Battle City. Group 2 will create a game in which tanks battle it out in top-down 2D graphics. The game is programmed in Java language, the available libraries and applied object-oriented programming thinking. The game's images and sounds were taken from opengameart.org. The program is completely designed and programmed by the group 2.

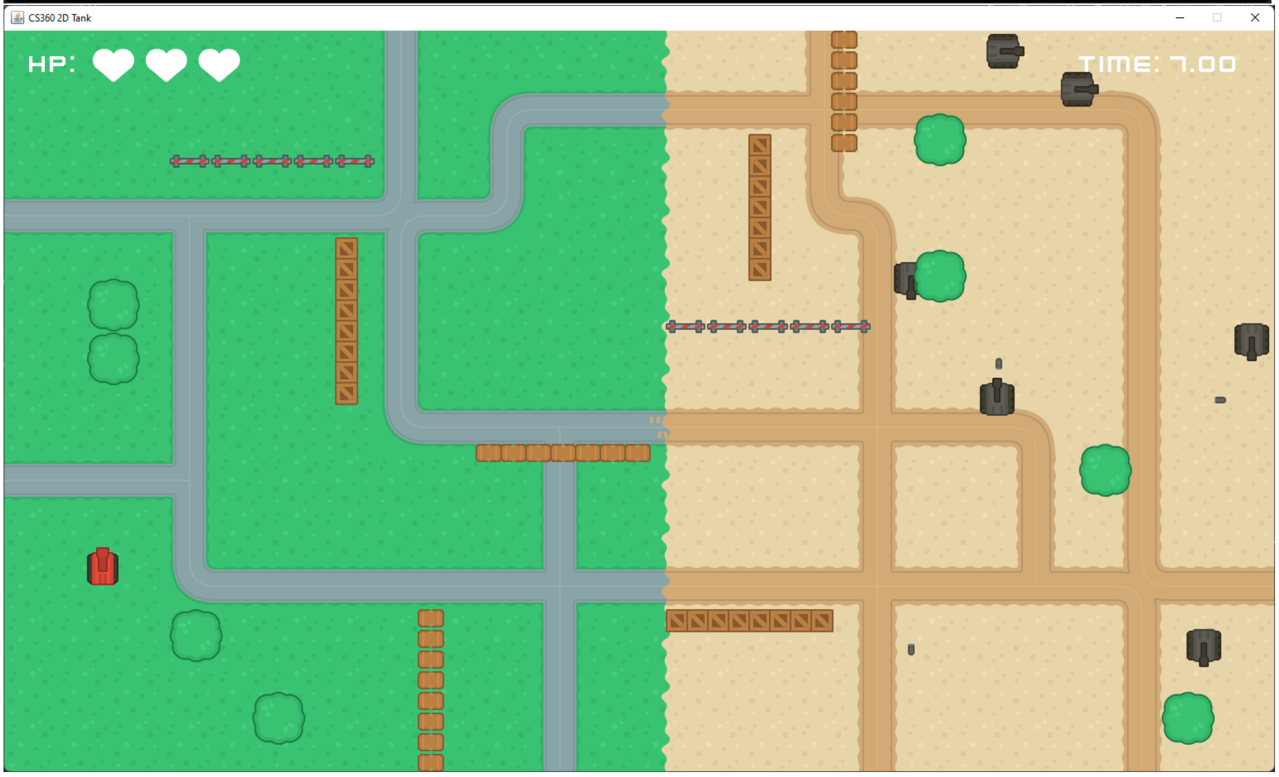

CC0 1.0
This content is free to use in personal, educational and commercial projects.
No need to ask permission. Support us by crediting, this is not mandatory.
KENNEY

II. Analysis and Program Design

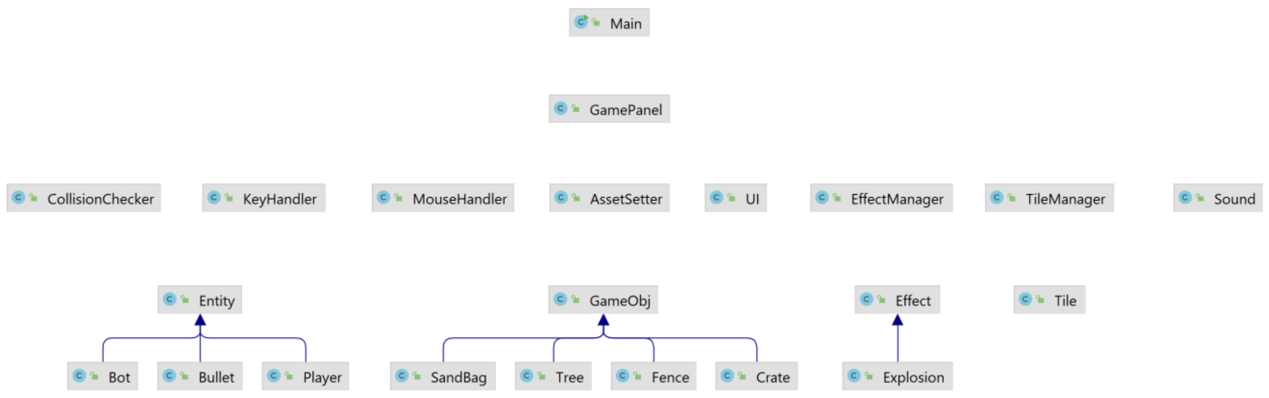
1. Feature Analysis

The game simulates a battle of tanks. The player controls his tank with the up, down, left, right navigation keys or the W, A, S, D keys. Players move in combination with the F key to shoot enemy tanks. Every tank has a health point, the tank will be destroyed when the health point is zero. In addition to the appearance of tanks, the map also has the presence of objects such as fences, sandbags, trees, and wooden crates. Objects can have different properties such as being destroyed or passed through or shot through. The player loses when destroyed and wins when all enemies are killed.





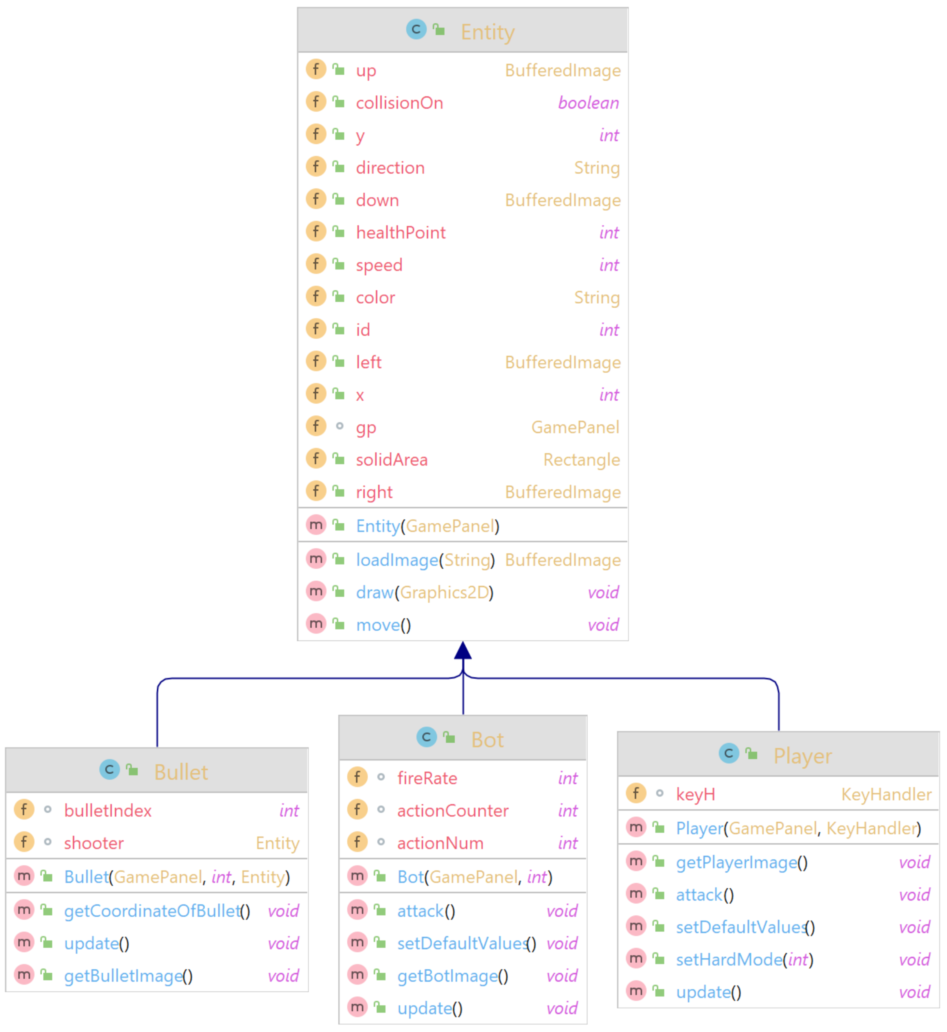
2. Class Diagram



3. Program Design

The program is divided into five packages: main, entity, effect, objet, and tile.

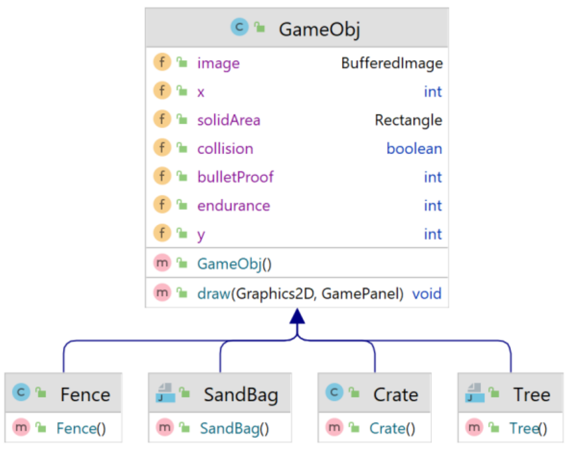
3.1 Package entity



The entity package contains four classes. Entity is the main class, and the other classes inherited from it are Bot, Player, and Bullet. The Entity class is designed to be a moving object. It has properties such as x and y to determine position, speed to determine movement speed, direction to determine movement direction, healthPoint to determine health point and collisionOn is the property that is enabled when the entity collides with other entities. The Entity class also has the solidArea property of the Retangle class, this variable is used when checking for collisions, and on collision, the collisionOn variable will be changed to true. Furthermore, the entity has images when moving in the up, down, left, and right directions. The Entity class has three primary methods: move, attack, and draw. Whereas the move method will update the object's position based on the direction, speed, and collisionOn. The draw method is used to draw an image of the entity on the screen; the method will select the image up, down, left, and right based on their orientation.

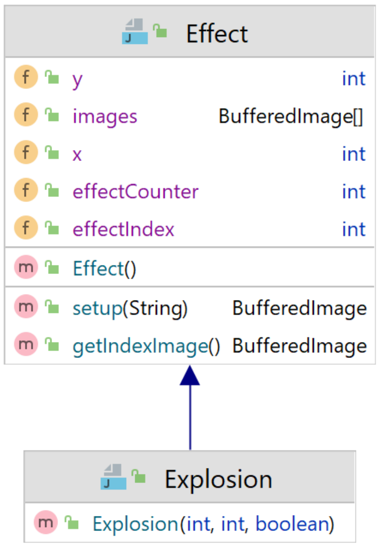
Inheriting the Entity class is the Bullet, Player, and Bot classes. Classes are added properties and methods to suit their functions. The Bot class will have features that are used to determine its direction, such as actionCounter, actionNum, and fireRate. In the Bot's update method, group 2 uses random as a straightforward way to let the bots make decisions. In addition, the Bot will have speed, healthPoint and fireRate based on the difficulty that the player chooses. With high difficulty, the Bot's tanks will run faster, shoot more bullets and be harder to destroy. The Player class is instantiated with an object of the KeyHandler class for the player to decide on the move and attack of the tank.

3.2 Package object



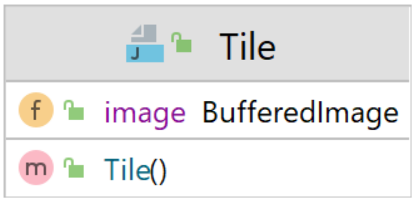
The object package has one main class, GameObj, and four classes inherited from GameObj: Fence, Crate, SandBag, and Tree. These are classes that represent the objects that will appear in the game. Like the Entity class, the GameObj also has some properties such as coordinates, the image of the BufferedImage class and the solidArea of the Retangle class that are used to check for collisions. In addition, they also have their own properties such as collision to show whether the object can be penetrated by tanks or not, bulletProof to show whether bullets can pass through or bullets can destroy, in in the case of destructible bullets, the endurance variable is so that the number of bullets the object can withstand. But the class inherited from GameObj is kept structurally but changed in value properties. Thereby making a difference between GameObj such as Fence that allows bullets to pass through but not tanks, SandBag cannot be destroyed but Crate will be destroyed after 5 bullets. The only method used in all these classes is the draw method used to draw their images onto the screen.

3.3 Package effect



This package includes two classes, Effect and Effect's inheritance class, Explosion. These are two simple classes with only four properties. The first is an array of BufferedImage named images, which is a collection of animations of an effect. Next are the coordinates, where the effect will be drawn. Finally, there are two int variables, effectCounter and effectIndex. When an effect is done, they will display each image in turn in a certain time, effectCounter helps us to count the number of times drawn of the current image, when enough time is enough, we will change the effectCounter to start drawing the next image. The Explosion class is simply an inherited class of the Effect class, and the difference is that when declaring it, it must declare a boolean smoke variable to decide whether it is a smoke explosion or not, and then define the pictures. image of the variable images.

3.4 Package tile

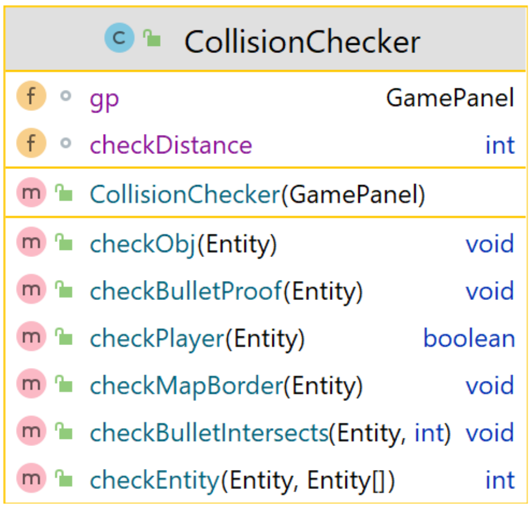


This tile package contains only one class, Tile. A BufferedImage variable is also the only property of the Tile class. This class is used to create the game's background tiles.

3.5 Package main

The final package is called main. This is the central package in full control of supporting, operating, and managing all the objects of the classes in the preceding packages. In this package, there are two main components of the game. The GamePanel class is extended from the JPanel class of the java.swing library and this is where most of the game's objects are initialized. Main class where the program starts is also the place to initialize the windows variable of the JFrame class and the gamePanel variable of the GamePanel class.

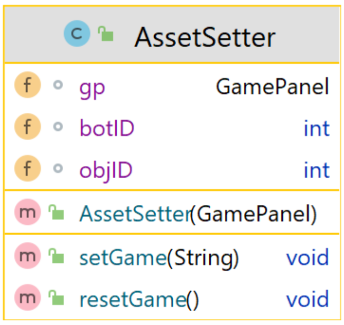
3.5.1 Class CollisionChecker


CollisionChecker
gp GamePanel
checkDistance int
CollisionChecker(GamePanel)
checkObj(Entity) void
checkBulletProof(Entity) void
checkPlayer(Entity) boolean
checkMapBorder(Entity) void
checkBulletIntersects(Entity, int) void
checkEntity(Entity, Entity[]) int

The CollisionChecker class is the part that helps the objects of the Entity class to check for collisions. There are two properties in this class, which are GamePanel gp and int checkDistance. The GamePanel gp variable is used to interact with objects running in the GamePanel. The checkDistance variable is the distance that this class will check before the collision happens when Entity executes the move method.

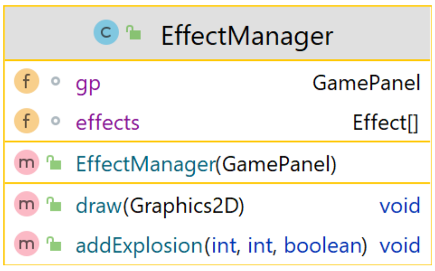
The Entity class owns a series of methods that are used to make objects of the Entity class check for collisions with map borders, other entities, and objects of the GameObj class. The methods all use a common mechanism that is to check in the future coordinates of an object of the Entity class that there is a collision or not, if so, it will change the collisionOn state of this object to true. With border the method will check for x, y coordinates with length, width and with 0 (map origin). When checking collisions with other GameObj and Entities, we will get the Retangle solidArea variable of the objects of the other classes, and then use the intersects method of the Retangle class to check the intersection between the two solidArea.

3.5.2 Class AssetSetter



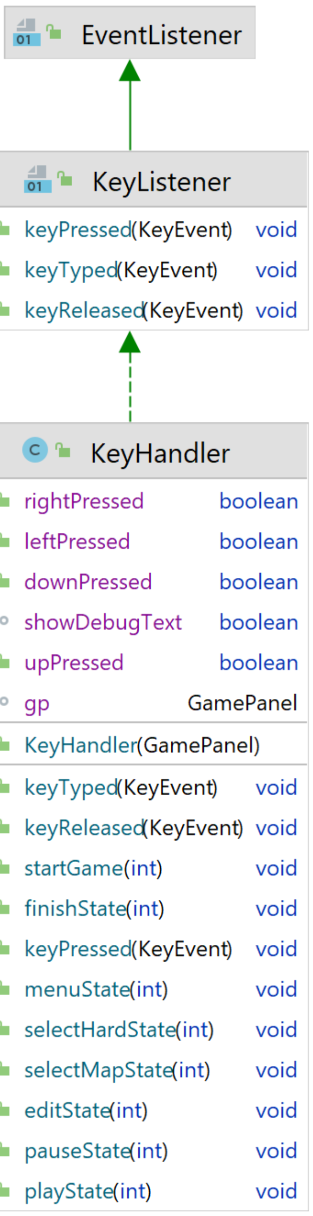
AssetSetter is a class used to set up and place GameObjs and Bot tanks on the map. This class has three properties: botID, objID and GamePanel gp. The main method of AssetSetter is setupGame. The setupGame method will load the data from the txt file and create the corresponding Obj in the GamePanel arrays. The txt files will have the id form to determine which class Obj needs to create and their coordinates. The variables botID and objID will increment respectively after each new obj is added to the array.

3.5.3 Class EffectManager



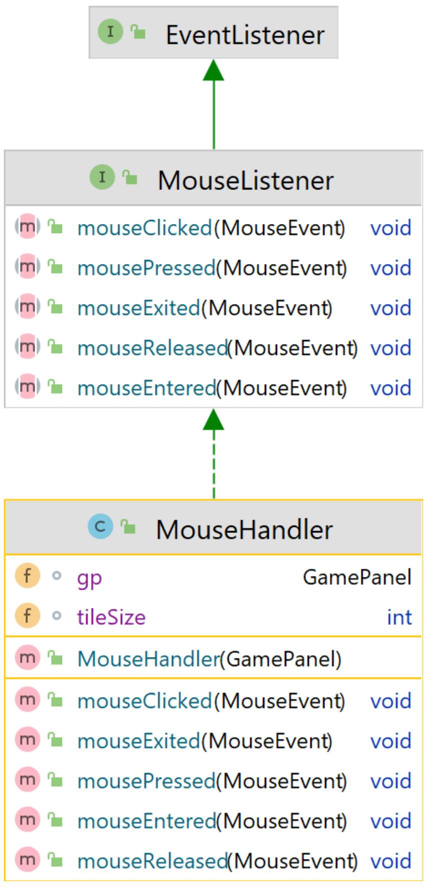
This class is where the array of the Effect class is stored. With the draw method of the EffectManager class will draw the animations of the effect one by one. Each image will be drawn in ten frames (10 loops). With the addExplosion method, EffectManager will create an explosion at a location specified in the parameter and whether it is of the smoke type or not.

3.5.4 Class KeyHandler



This is a class to receive pressed keys and execute its corresponding function. This class is based on two classes of the awt library, KeyEvent and KeyListener. With different gameStates, there will be different sets of keys and corresponding functions.

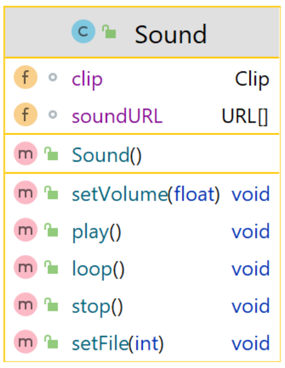
3.5.5 Class MouseHandler



EventListener
MouseListener
mouseClicked(MouseEvent) void
mousePressed(MouseEvent) void
mouseExited(MouseEvent) void
mouseReleased(MouseEvent) void
mouseEntered(MouseEvent) void
MouseHandler
gp GamePanel
tileSize int
MouseHandler(GamePanel)
mouseClicked(MouseEvent) void
mouseExited(MouseEvent) void
mousePressed(MouseEvent) void
mouseEntered(MouseEvent) void
mouseReleased(MouseEvent) void

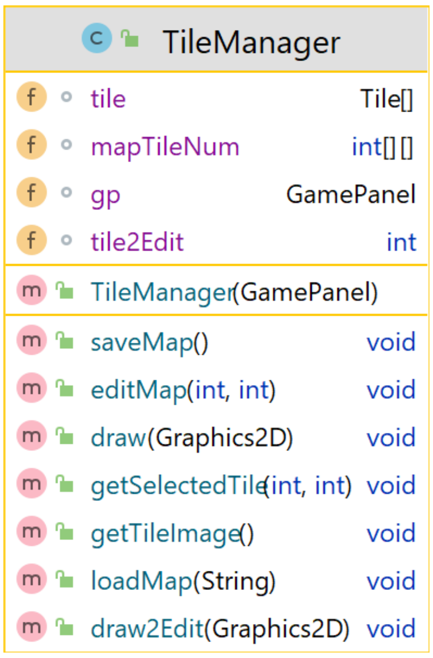
Like KeyHandler, MouseHandler is used to receive mouse events based on two classes of the awt library, MouseListener and MouseEvent. In the current version MouseHandler is only used when the user edits the map. Every time the user clicks the mouse, it will call a method from the TitleManager, through the coordinates given by the MouseHandler, the TileManager will determine the location where the user clicked the mouse.

3.5.6 Class Sound



Class Sound is the class used to play the background music as well as the sound effects of the game. The class is built on top of classes like AudioInputStream, AudioSystem, Clip, FloatControl belonging to javax.sound.sampled. The class has two main properties, the clip of the Clip class and a soundURL array of the URL class. After the constructor of the Sound class is initialized, it loads the URLs of the sounds to be used. And then it will be set up with the setFile method to select the music, the setVolume method to adjust the sound. Finally, the classes play, loop, and stop to stop, play once, or loop that sound.

3.5.7 Class TileManager



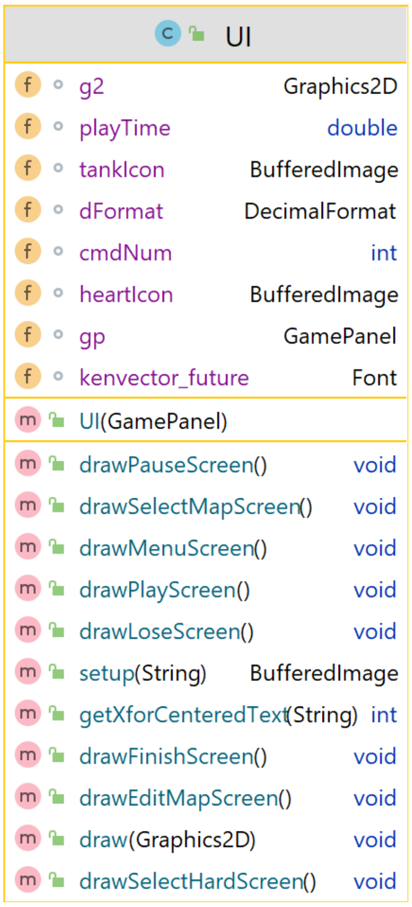
```
TileManager
tile                        Tile[]
mapTileNum                  int[][]
gp                          GamePanel
tile2Edit                   int
TileManager(GamePanel)
saveMap()                   void
editMap(int, int)           void
draw(Graphics2D)            void
getSelectedTile(int, int)   void
getTileImage()              void
loadMap(String)             void
draw2Edit(Graphics2D)       void
```

The TileManager class is the main class to manage the Tiles in the background of the level and provides methods for modifying the map. The core of TileManager is that it is a 2D matrix of int variables. Then will draw the corresponding Tiles on the user's play screen. In addition to that 2-dimensional matrix, there are three other properties that are GamePanel gp to interact with the main Gamepanel, tile array of Tile class with size forty to store forty different types of Tiles and an int variable called tile2Edit to use. while editing the map.

Starting with the constructor, the initializer will load the image files into the array and will initialize the size of the 2D matrix to match the game's config. Then with the loadMap method reads the data of the mapTile matrix as a txt file. TileManager's draw method, as well as that of other classes, will draw Tiles based on a 2D mapTile matrix. In both the loadMap and draw methods, the main component is the loop to go through the 2-dimensional matrix.

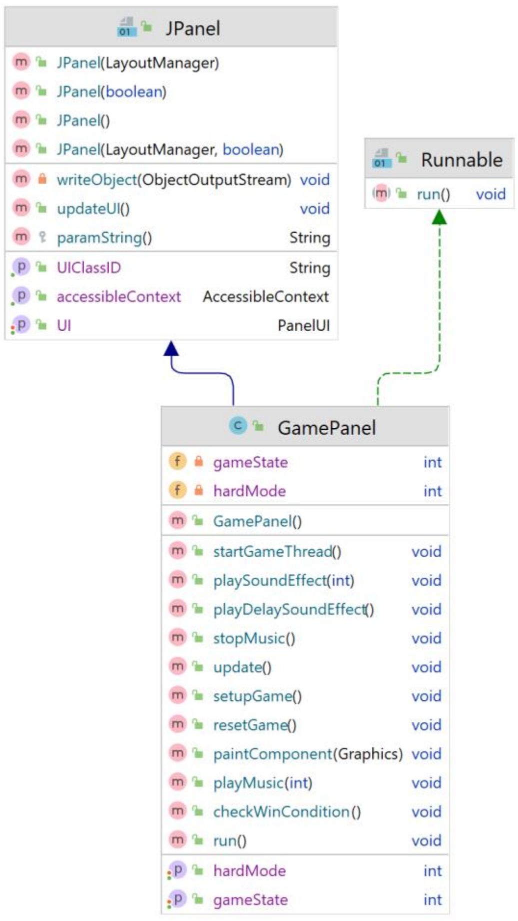
In addition to the main methods above, TileManager also has methods used to edit the map. Start with the draw2Edit method to draw 40 Tiles for the user to select into the current map matrix for the user to see the change. The two methods getSelectedTile and editMap are two methods that are called from MouseHandler. These two methods will select a tile from forty tiles or change the current map depending on the position of the mouse. And finally, the saveMap method to save the recent changes to the txt file.

3.5.8 Class UI



This is a class to display the user interface depending on the state of the game. With two main properties, Gamepanel gp to connect to Gamepanel and Graphics2D g2 to draw interfaces. The cmdNum property to represent the user's choices in menus. Double playTime property to display the time of the match. There are also properties to support drawing interfaces such as Font kenvector_future, BufferedImage heartIcon, BuffredImage tankIcon, DecimalFormat dFormat. Thereby with the drawing methods corresponding to the state of the game, the UI class will draw the image with the appropriate content.

3.5.9 Class Gamepanel



The Gamepanel class is the main class of the game where all the remaining classes are stored, and the game loop of the game is stored. This class is inherited from the JPanel class and the Runnable imlements to push to the Threads.

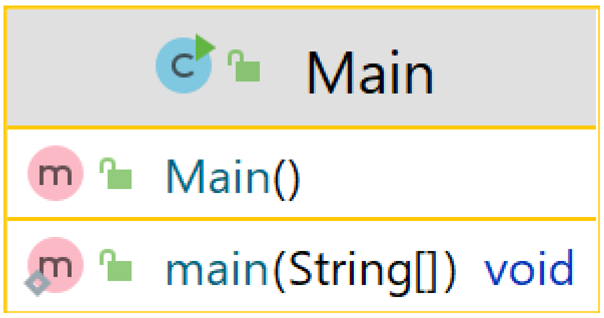
At the beginning are variables that represent the game's config such as tileSize to indicate the size of each Tile (here, sixty-four is equivalent to a Tile of size sixty-four * 64), maxScreenCol and maxScreenRow are the row and column numbers of mapTile, multiplying those two together we get screenWidth and screenHigh in terms of pixels and FPS of the game. Next is the config variable gameState and values such as menuState, playState, pauseState, winState, loseState, editState, selectMapState, selectHardState. Constant values will be assigned to the gameState so that the classes will behave according to the current gameState of the game. Change hardMode to set the difficulty of the game. The two variables mapPath and objPath are directed to the two default txt files of map.

After the game's config variables will be the variables to operate the game. The first is the gameThread variable of the Thread class, the gameThread will contain this GamePanel and then it will run the start method. Next will initialize the classes introduced above such as KeyHandler keyH, MouseHandler mouseH, two variables of the Sound class are music and effects, UI ui, EffectManager effectM, TileManager tileM, CollisionChecker cChecker, AssetSetter aSetter, Player player, array bot of class Bot, bullet array of class Bullet, array obj of class GameObj. Finally, there are variables used to support other features such as boolean delayOn, int delayCounter, boolean endSound for serving audio, systemIsMacOS variable for serving loadMap of different operating systems.

The GamePanel constructor here sets up the states for the JPanel such as setPreferredSize, setBackground, setDoubleBuffered, setFocusable and addKeyListener along with addMouseListener. Coming to the first method called setupGame, a simple method that determines if the operating system is MacOS, loads the map, plays the background music, and sets the gameState to menuState.

The GamePanel constructor here sets up the states for the JPanel such as setPreferredSize, setBackground, setDoubleBuffered, setFocusable and addKeyListener along with addMouseListener. Coming to the first method called setupGame, a simple method that determines if the operating system is MacOS, loads the map, plays the background music, and sets the gameState to menuState. The resetGame method refreshes the game data variables so that some new games can be played. The run method is an overwritten method, in the run method will contain the game loop, the game will call two methods update and repaint every 1/60s (FPS = 60), these two methods are inherited from JPanel. However, the update method will be redefined, when the gameState is in the playState the game will call the update methods of the player, bot, and bbullet. With paintComponent method also rewritten to call all draw methods coming from above classes. Next is the checkWinCondition method, this method will be called by the objects of the Bullet class every time a Bot is shot dead, when there are no more Bots, the method will change the gameState to winState. Finally, there are methods for playing sound like playMusic, stopMusic, playSoundEffect, playDelaySoundEffect and some get set methods for gameState and hardMode.

3.5.10 Class Main



The last class is the Main class where the program starts, here is where the JFrame and JPanel (rewritten as GamePanel) are defined and launched.

Contribution Summary Table:

Name	Task
Pham Tran Tri	Tile, TileManager
Doan Dinh Cuong	UI, Sound, KeyHandler, MouseHandler
Ngo Truong Minh	GameObj, AssetSetter, Effect, EffectManager
Dao Trung Hieu	GamePanel, Programming Design
Nguyen Trung Kien	Entity, CollisionChecker

Source code:

https://github.com/hieutrungdao/CS360_2DTank

References:

1. Game assets: https://opengameart.org/content/top-down-tanks-redux