|  |
| --- |
| Champlain Regional College |
| Computer Science and Mathematics Final Project |
| Phase 1: Design of TANKS |

|  |
| --- |
| William Adam-Grenier & Cedrik Dubois  2-26-2017 |

Table of Contents

**Introduction2**

Story of the Game2

Game3

**Design3**

Analyze Problem3

Algorithm4

UML……………………………………………………………………………………………………………………………………………..7

UML Description……………………………………………………………………………………………………………………….8

GUI & Character Design……………………………………………………………………………………………………….25

Timeline…………………………………………………………………………………………………………………………..28

Tools Used to Create Program……………………………………………………………………………………..29

**Features30**

**Conclusion33**

**Introduction**

**Story of the Game**

          The story of this game is not a very hard one to follow. Here we are in 2050 where the world is probably coming to an end very soon. North Korea under the ruling of Kim Jong Un was finally able to develop and mass produce full functioning nuclear weapons. With China as their only ally, they planned to attack the United States of America when they mistakenly landed their troops in Canada, more precisely on the shores of Newfoundland and Labrador. Canada saw this as a threat naturally and with very little time to think decided retaliate by attacking the incoming troops. After this decision, the US was not given much of a choice but to help Canada in their war against North Korea. As the US started to mobilize troops, China’s intelligence got news of this and decided to do the same for their ally. So for months, Canada and the US fought against North Korea and China while the rest of the world watched, until that one day. It was lunch time so everyone stopped fighting to have something to eat. As they were eating, one of the Canadian soldiers decided to go take a quick nap in his tank, but as he was getting in, stepped on the fire button and launched a shell, hitting a fellow American tank. The Canadian kept apologizing over and over as they usually do, but the Americans were having none of it. They also fire a shell back, but miss, and hit a tank on the other side of the hill, which happened to be a Chinese tank. Since they were on their lunch break too, they didn’t see where the shot came from, and thought that North Korea was probably turning on them. And so that was the day where everyone turned on each other. It was that day that four countries went in a free-for-all. We call that day “the incident”, and you my friend are a soldier that is part of “the incident”, and you have no one but yourself and your tank to count on. Your only goal, be the last one standing.

**Game**

When we first started this project, one thing was clear, we wanted to make a game. However, it couldn’t be any game since we needed to have either a math, physic or chemistry component to it, and that is how we thought of the game Tanks. This game is based off a very old and popular game called Worms, where friends can fight each other and the last one to survive wins. Tanks is very similar to this except we wanted our players to learn a thing or two about our physical world. This game not only lets you battle with friends but teaches the players about some of the most fundamental properties of nature, and that is the laws of motion. The goal of the game is to hit other players on the map with a variety of projectiles such as missiles and shrapnel rounds. The difficulty of the game is that most of these weapons are being affected by properties of nature like gravity and wind resistance.  This means that players will need to learn how these properties affect launched projectiles and adapt to the weather conditions of the map. By doing this we hope that the users learn something about the laws of motion and how the physical world of nature works. One big feature players will need to adjust to be wind resistance, as that can change at any point during the game, and has a strong impact on objects high in the air.

          Our objective when building this game is to develop a fun and interactive way for users to understand physics, and a little of math too. Most classes on the subject are very boring and don’t grasp student’s attention very well, however when you play this game, you learn without realizing. Obviously, this game would target a much younger audience, but just the fact that players need to learn to shoot with or against the wind is enough to get their brains going. Having little vs a lot of gravity also teaches the important role it plays on earth.

**Design**

**Analyze Problem**

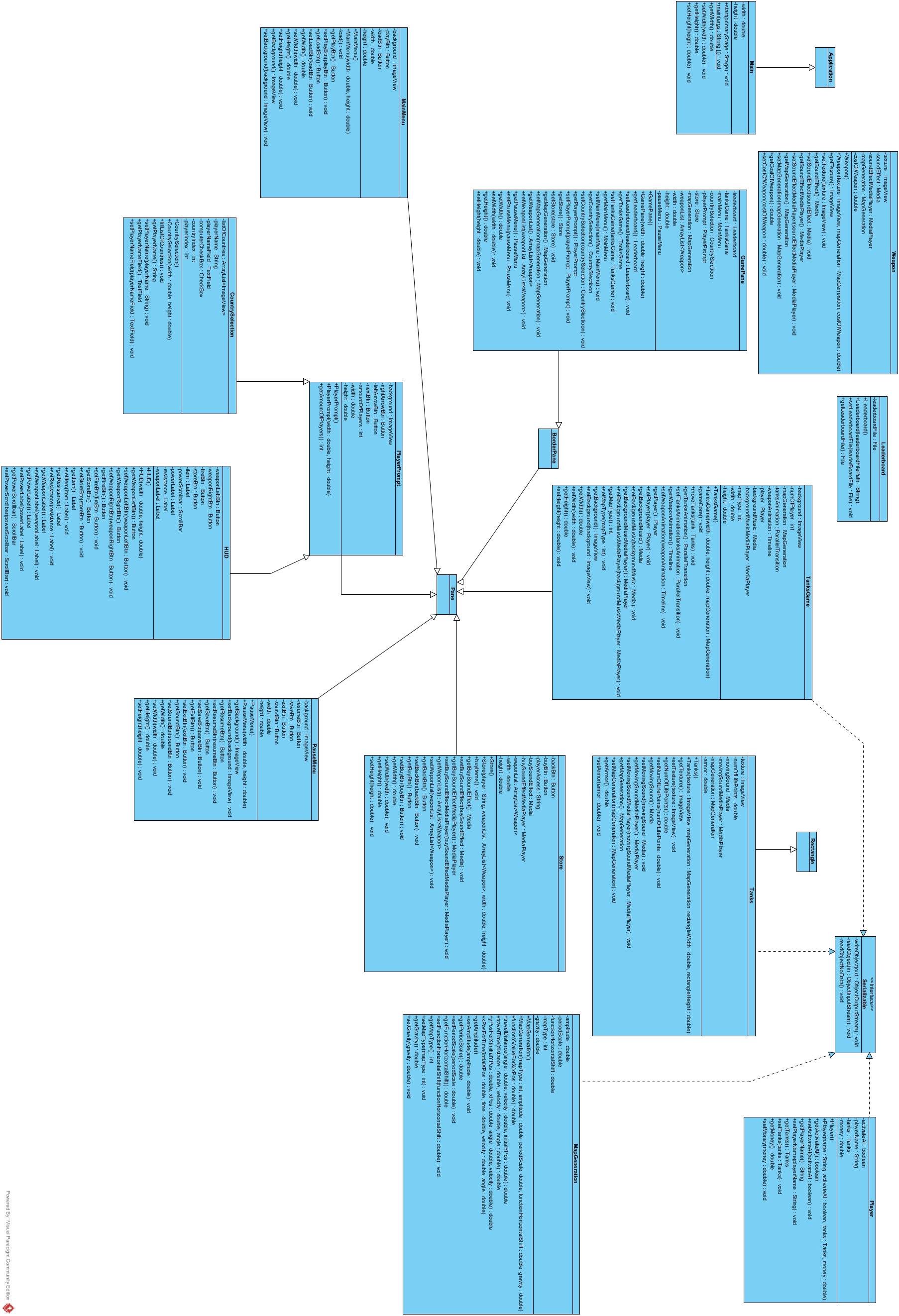
The game must be simple to understand and easy to play. The game utilizes two-dimensional (2D) gameplay to allow for an easy learning curve for new gamers. Also, 2D graphics does not have the same time-consuming requirement of 3D graphics. Therefore, the team can focus all of its potential to create a unique and fun experience to gamers. All UI elements must be intuitive and organized into different categories. Thus, the look and feel of old retro games inspired this project to make the user feel at home. Up to 4 tanks are competing on a map to win the game. The last one to stand wins the game. To do so, the players have at their disposal a vast array of weapons with simulated physics for air motion. The physics of the weapon will be calculated with the motion equations and for the variables X and Y. The tanks can either be controlled by humans or by an AI. This features allow the game to be versatile and playable in any situation: AI vs Human or Human vs Human. The maps are generated by a sine function to create a more realistic feel to the game. Based on the location of the map, which can either be desert, hills, forest, and an unknown planet, will have different properties for their unique location. For example, the forest map will have less hills than the mountain map. These features are going to be the hardest parts to implement in the program because of the complexity of the equations and to incorporate them into the game. The program also includes a save/load function and a local leaderboard to make a more continuous experience to the user. These functions will be written in a binary file on the computer and accessed when needed.

**Algorithm**

First, the formulas for the simulation of ballistics will be used to calculate the precise trajectory and time of the weapon to an animation of the type Timeline. The results will be animations with a realistic feel to our world’s physics. All objects in the game that can be thrown into the air will have its trajectory calculated. The total horizontal distance traveled by an object is calculated with the formula ). The v in the formula represents the velocity at which the projectile was launched. G represents the gravity, or the downward force that the projectile feels. The higher the variable G is, the faster the object will hit the ground. Yi is the initial height at which the projectile was launched. is the angle (in degrees) that the projectile was launched. This formula will be used to calculate where the projectile will land on the horizontal plane. The efficiency of this algorithm is O(1) because the calculation is done only once before launching the projectile and returns a constant. To calculate to total time of the flight, the formula and will be used to determine the length in time of the animation. The time of the flight is the distance divided by the rate at which the projectile travels. For the variable X, or the horizontal displacement, the initial velocity will equal to the final velocity, Vxi = Vxf. Therefore, this equation will only be calculated once before the start of the animation. The efficiency of this algorithm is O(1) because it is constant calculated once. To calculate the y position of the projectile at any given x distance, the formula is used. The formula will be used during the displacement of the projectile in the air to find the accurate y-location. The efficiency of this algorithm is O(n), where n represents the number of pixel to traverse. This formula is used to calculate the next y-position of the projectile. Thus, the efficiency is directly related to the number of pixels to traverse. To find the x-position of the projectile, the formula is used. In this case, the variable t represents the amount of time passed since the projectile was launched. Xi is the initial x-position of the projectile when launched. The efficiency of this algorithm is O(n), for the same reasons as the y-position algorithm.

Second, the map generation will use the sine function to create adequate terrain to the current map style. The general formula of a sine function is . The variable a represents the amplitude of the function, or the maximum y value of the function. This variable will depend on the map. For example, the desert maps will have a low amplitude because there is not a lot of variation of the terrain. For the snowy valley maps, the amplitude will be high because the location has a high variation of ground levels. The variable K represents the period of the function, or the rate at which one full period will be done. The higher the period is, the less frequent the hills will be. The variable X represents a pixel in the total width of the game’s pane. The variable b represents the displacement of the initial value when x equals zero. This creates a shift towards the left or the right for the hill. This allows to put a hill in the middle of the map to create a new style of gameplay. The efficiency of the algorithm is O(n) because the loop for creating the terrain will depend on the width of the pane. The loop will be traversed the same number of times as the number of pixels in the width.

**UML**



**UML Description**

Main extends Application

Class Description

This is the main class. The application is launched from here.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| Private double | height  used to set the pane’s height |
| Private double | width  used to set the pane’s width |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public void | start(primaryStage: Stage)  called to start the application |
| public static void | main(args: String[])  main method used to launch the program |
| public double | getWidth()  returns the double value of width |
| public void | setWidth(double width)  sets the value of width to the new value |
| public double | getHeight()  returns the double value of height |
| public void | setHeight(double height)  sets a new value for the height |

Weapon

Class Description

The class weapon is for creating the in-game weapons that will be used to destroy the tanks.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private ImageView | texture  the texture of that weapon |
| private Media | soundEffect  the sound played when this weapon is used |
| private MediaPlayer | soundEffectMediaPlayer  the player used to play the sound effect |
| private MapGeneration | mapGeneration  used to generate the map and find coordinates on it |
| private double | costOfWeapon  variable that stores the value of that weapon |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | Weapon()  default constructor of Weapon |
| public | Weapon(ImageView texture, MapGeneration mapGeneration, double costOfWeapon )  the constructor of Weapon with non-default values |
| public ImageView | getTexture()  returns the ImageView of the texture |
| public void | setPicture(ImageView texture)  sets a new ImageView for the texture |
| public Media | getSoundEffect()  returns the media for the sound effect |
| public void | setSoundEffect(Media soundEffect)  sets a new Media for the sound effect |
| public MediaPlayer | getSoundEffectMediaPlayer()  returns the MediaPlayer for the sound effect |
| public void | setSoundEffectMediaPlayer(MediaPlayer soundEffectMediaPlayer)  sets a new MediaPlayer for the sound effect Media Player |
| public MapGeneration | getMapGeneration()  returns the mapGeneration |
| public void | setMapGeneration(MapGeneration mapGeneration)  sets a new MapGeneration for the mapGeneration |
| public double | getCostOfWeapon()  returns the cost of the weapon |
| public void | setCostOfWeapon(double costOfWeapon)  sets a new cost for the weapon |

Leaderboard

Class Description

The leaderboard contains the best 10 players of the game. It writes the name of the player and next to it the score in a file of the format .txt.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private File | leaderboardFile  a file that stores the leaderboard |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | Leaderboard()  default leaderboard constructor |
| public | Leaderboard(String leaderboardFilePath)  the leaderboard constructor for a different file path |
| public void | setLeaderboardFile(File leaderboardFile)  sets a new file for the leaderboard |
| public File | getLeaderBoardFile()  returns the file of the leaderboard |

GamePane extends BorderPane

Class Description

GamePane is the main pane of the game containing all of the required elements in order to play the game.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private Leaderboard | leaderboard  variable for storing the leaderboard |
| private TanksGame | tanksGame  variable for storing the tanks game pane |
| private MainMenu | mainMenu  variable for storing the main menu pane |
| private CountrySelection | countrySelection  variable for storing the country selection pane |
| private PlayerPrompt | playerPrompt  variable for storing the player Prompt pane |
| private Store | store  variable for storing the store pane |
| private MapGeneration | mapGeneration  used to generate the map and find coordinates on it |
| private ArrayList<Weapon> | weaponList  variable to store all of the different weapon types |
| Private double | height  used to set the pane’s height |
| Private double | width  used to set the pane’s width |
| private PauseMenu | pauseMenu  variable for storing the pause menu pane |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | GamePane()  default no arg constructor |
| public | GamePane(double width, double height)  Constructor that takes in the width and the length of the pane |
| public Leaderboard | getLeaderboard()  returns the leaderboard |
| public void | setLeaderboard(Leaderboard leaderboard)  sets a new leaderboard |
| public TanksGame | getTanksGame()  returns the tanks game pane |
| public void | setTanksGame(TanksGame tanksGame)  sets a new tanks game pane |
| public MainMenu | getMainMenu()  returns the main menu |
| public void | setMainMenu(MainMenu mainMenu) sets a new main menu pane |
| public CountrySelection | getCountrySelection()  returns the country selection |
| public void | setCountrySelection(CountrySlection countrySelection) sets a new country selection pane |
| public PlayerPrompt | getPlayerPrompt()  returns the player Prompt |
| public void | setPlayerPrompt(PlayerPrompt playerPrompt) sets a new player prompt pane |
| public Store | getStore()  returns the store |
| public void | setStore(Store store) sets a new store pane |
| public | get()  returns the |
| public void | set sets a new |
| public MapGeneration | getMapGeneration()  returns the mapGeneration |
| public void | setMapGeneration(MapGeneration mapGeneration)  sets a new MapGeneration for the mapGeneration |
| public ArrayList<Weapon> | getWeaponList()  returns weaponList |
| public void | setWeaponList(ArrayList<Weapon> weaponList) sets a new weaponList |
| public PauseMenu | getPauseMenu()  returns the pauseMenu pane |
| public void | setPauseMenu(PauseMenu pauseMenu) sets a new pause menu |
| public double | getWidth()  returns the double value of width |
| public void | setWidth(double width)  sets the value of width to the new value |
| public double | getHeight()  returns the double value of height |
| public void | setHeight(double height)  sets a new value for the height |

MainMenu extends Pane

Class Description

The main menu pane is the pane launched before starting the game. It lets the user decide to start a new game or load a previous one.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private ImageView | background  an image background for the pane |
| private Button | playBtn  a button to launch the game |
| private Button | loadBtn  a button to load a previous save |
| Private double | height  used to set the pane’s height |
| Private double | width  used to set the pane’s width |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | MainMenu()  A default no-args constructor for MainMenu |
| public | MainMenu(double width, double height)  Constructs object with defined width and height |
| private void | load()  Loads a previous save of the game |
| public Button | getPlayBtn()  returns the play button |
| public void | setPlayBtn(Button playBtn) sets a new play button |
| public Button | getLoadBtn()  returns the load button |
| public void | setLoadBtn(Button btn) sets a new load button |
| public ImageView | getBackground()  returns the image of the background |
| public void | setBackground(ImageView background) sets a new background |
| public double | getWidth()  returns the double value of width |
| public void | setWidth(double width)  sets the value of width to the new value |
| public double | getHeight()  returns the double value of height |
| public void | setHeight(double height)  sets a new value for the height |

CountrySelection extends PlayerPrompt

Class Description

The country selection pane is where the user selects the country to play with and the number of players. The user also enters the name here.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private ArrayList<ImageView> | listOfCountries  The list of all the countries |
| private TextField | playerNameField  TextField to enter the name of the player |
| private String | playerName  String to store the player’s name |
| private CheckBox | computerCheckBox  CheckBox to active the AI |
| private int | playerIndex  Stores the index of the Player |
| private int | countryIndex  Stores the index of the country |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | CountrySelection()  Default no-arg constructor |
| public | CountrySelelction(double width, double height)  Constructs object with defined width and height |
| public void | fillListOfCountries()  Fills the list with the countries |
| public String | getPlayerName()  Return the String of the player’s name |
| public void | setPlayerName(String playersName)  sets a new name for the player |
| public TextField | getPlayerNameField()  Returns the text field of the player’s name |
| public void | setPlayerNameField(TextField playerNameField)  Sets a new TextField for the player’s name |

PlayerPrompt extends Pane

Class Desciption

The class contains the basic prompt buttons and background to interact with the player.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private ImageView | background  an image background for the pane |
| private Button | leftArrowBtn  Button for moving left |
| private Button | rightArrowBtn  Button for moving right |
| private Button | nextBtn  Button for going to the next pane |
| private int | amountOfPlayers  Sets the number of players in the game |
| Private double | height  used to set the pane’s height |
| Private double | width  used to set the pane’s width |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | PlayerPrompt()  Default no-arg constructor |
| public | PlayerPrompt(double width, double height)  Constructs object with defined width and height |
| public ImageView | getBackground()  returns the image of the background |
| public void | setBackground(ImageView background) sets a new background |
| public Button | getRightArrowBtn()  returns the right arrow button |
| public void | setRightArrowBtn(Button rightArrowBtn) sets a new right arrow button |
| public Button | getLeftArrowBtn()  returns the left arrow button |
| public void | setLeftArrowBtn(Button leftArrowBtn) sets a new left arrow button |
| public Button | getNextBtn()  returns the next button |
| public void | setNextBtn(Button nextBtn) sets a new next button |
| public double | getWidth()  returns the double value of width |
| public void | setWidth(double width)  sets the value of width to the new value |
| public double | getHeight()  returns the double value of height |
| public void | setHeight(double height)  sets a new value for the height |

HUD extends PlayerPrompt

Class Description

The HUD is the pane that gives the player information about the status of the tank and its inventory.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private Button | weaponLeftBtn  Button for changing to next weapon |
| private Button | weaponRightBtn  Button for changing to previous weapon |
| private Button | fireBtn  Button for firing weapon |
| private Button | storeBtn  Button that opens the store pane |
| private Label | item  Shows the name of the current item |
| private ScrollBar | powerScrollbar  Scrollbar to set strength to throw weapon |
| private Label | powerLabel  label to show power to throw weapon |
| private Label | resistance  Shows the current resistance |
| private Label | weaponLabel  Shows the name of the current weapon |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | HUD()  Default no-arg constructor |
| public | HUD(double width, double height)  Constructs object with defined width and height |
| public Button | getWeaponLeftBtn()  returns weaponLeftBtn |
| public void | setWeaponLeftBtn(Button weaponLeftBtn) sets a new weaponLeftBtn |
| public Button | getWeaponRightBtn ()  returns the weaponRightBtn |
| public void | setWeaponRightBtn(Button weaponRightBtn) sets a new weaponRightBtn |
| public Button | getFireBtn ()  returns the fireBtn |
| public void | setFireBtn(Button fireBtn) sets a new fireBtn |
| public Button | getStoreBtn ()  returns the storeBtn |
| public void | setStoreBtn(Button storeBtn) sets a new storeBtn |
| public Label | getItem()  returns item |
| public void | setItem(Label item) sets a new item label |
| public Label | getResistance()  returns the resistance |
| public void | setResistance(Label resistance sets a new resistance |
| public Label | getWeaponLabel()  returns the weapon label |
| public void | setWeaponLabel(Label weaponLabel) sets a new weapon label |
| public Label | getPowerLabel()  returns the power label |
| public void | setPowerLabel(Label powerLabel) sets a new power label |
| public Scrollbar | getPowerScrollbar()  returns the power scroll bar |
| public void | setPowerScrollbar(ScrollBar powerScrollbar) sets a new power scroll abr |

PauseMenu extends Pane

Class Description

The pause menu pane is the pane that appears to the user when the game is set to pause. During this time, the game is frozen.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private ImageView | background  an image background for the pane |
| private Button | resumeBtn  Button to resume the game |
| private Button | saveBtn  Button to save the game’s progreass |
| private Button | exitBtn  Button to exit the game |
| private Button | soundBtn  Button to stop the music |
| private double | height  used to set the pane’s height |
| private double | width  used to set the pane’s width |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | PauseMenu()  Default no-arg constructor |
| public | PauseMenu(double width, double height)  Constructs object with defined width and height |
| public ImageView | getBackground()  returns the image of the background |
| public void | setBackground(ImageView background) sets a new background |
| public Button | getResumeBtn()  Returns the resume Button |
| public void | setResumeBtn(Button resumeBtn)  Sets a new resume Button |
| public Button | getSaveBtn()  Returns the save Button |
| public void | setSaveBtn(Button saveBtn)  Sets a new save Button |
| public Button | getExitBtn()  Returns the exit Button |
| public void | setExitBtn(Button exitBtn)  Sets a new exit Button |
| public Button | getSoundBtn()  Returns the sound Button |
| public void | setSoundBtn(Button soundBtn)  Sets a new sound Button |
| public double | getWidth()  returns the double value of width |
| public void | setWidth(double width)  sets the value of width to the new value |
| public double | getHeight()  returns the double value of height |
| public void | setHeight(double height)  sets a new value for the height |

Store extends Pane

Class Description

The store pane is the pane that contains all of the weapons available to the players.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private Button | backBtn  Button to go back to previous pane |
| private String | playerAccess  String for the player’s name in store |
| private Button | buyBtn  Button to buy the item |
| private Media | soundEffect  Sound made when item is purchased |
| private MediaPlayer | buySoundEffectMediaPlayer  Player for the buy sound effect |
| private ArrayList<Weapon> | weaponList  ArrayList of weapon containing all weapons |
| private double | height  used to set the pane’s height |
| private double | width  used to set the pane’s width |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | Store()  Default no-arg constructor |
| public | Store(String player, ArrayList<Weapon> weaponList, double width, double height)  Constructor with args for a player name, the list of weapons, and the width and height of pane |
| public void | buyItems()  When the user buys something in the store |
| public Media | getBuySoundEffect()  returns the buy sound effect media |
| public void | setBuySoundEffect(Media buySoundEffect) sets a new sound effect when buying |
| public MediaPlayer | getBuySoundEffectMediaPlayer()  returns the media player for the buy sound effect |
| public void | setBuySoundEffectMediaPlayer(MediaPlayer buySoundEffectMediaPlayer) sets a new media player for the buy sound effect media player |
| public ArrayList<Weapon? | getWeaponList()  returns the weapon list |
| public void | setWeaponList(ArrayList<Weapon> weaponList) sets a new weapon list |
| public Button | getBackBtn()  Returns the back Button |
| public void | setBackBtn(Button backBtn)  Sets a new back Button |
| public Button | getBuyBtn()  Returns the buy Button |
| public void | setBuyBtn(Button buyBtn)  Sets a new buy Button |
| public double | getWidth()  returns the double value of width |
| public void | setWidth(double width)  sets the value of width to the new value |
| public double | getHeight()  returns the double value of height |
| public void | setHeight(double height)  sets a new value for the height |

<<Interface>>

Serializable

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| private void | writeObject(ObjectOutputStream out) |
| private void | readObject(ObjectInputStream in) |
| private void | readObjectNoData() |

TanksGame extends Pane implements Serializable

Class Description

This class is where the game mechanics are controlled. The main loop of the game is situated in the method gameCore(). The tanks and the map are situated in this pane. When the game is saved, this class is written into a file.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private int | numOfPlayer  Used to set how many players in the game |
| private MapGeneration | mapGeneration  used to generate the map and find coordinates on it |
| private ParallelTransition | tanksAnimation  Animation for the tanks |
| private Timeline | weaponAnimation  Animation for throwing weapon |
| private Player | player  Players related to this game |
| private Media | backgroundMedia  Background music of the game |
| private MediaPlayer | backGroundMusicMediaPlayer  Media player for the background music |
| private ImageView | background  an image background for the pane |
| private int | mapType  Sets the type of map |
| private double | height  used to set the pane’s height |
| private double | width  used to set the pane’s width |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | TanksGame()  Default no-arg constructor |
| public | TanksGame(MapGeneration mapGeneration, double width, double height)  Constructor to set pane with width and height and a map to generate |
| private void | gameCore()  This method is called internally to start the game’s main loop. When this ends, the game ends |
| public void | moveTanks(Tanks tank)  Move this tank to the specified location |
| public ParallelTransition | getTanksAnimation()  returns the tanks animation |
| public void | setTanksAnimation(ParallelTransition tanksAnimation)  Sets a new animation for the taks |
| public Timeline | getWeaponAnimation()  Returns the weapon animation |
| public void | setWeaponAnimation(Timeline weaponAnimtion) Sets a new animation for a weapon |
| public Player | getPlayer()  Returns the player |
| public void | setPlayer(Player player) Sets a new player |
| public Media | getBackgroundMusic()  Returns the background music media |
| public void | setBackgroundMusic(Media backgroundMusic) Sets a new background music |
| public MediaPlayer | getBackgroundMusicMediaPlayer()  Returns the background Music Media Player |
| public void | setBackgroundMusicMediaPlayer(MediaPlayer backgroundMusicMediaPlayer) Sets a new media player for the background player |
| public int | getMapType()  Returns the map type |
| public void | setMapType(int mapType) Sets a new mapType |
| public ImageView | getBackground()  returns the image of the background |
| public void | setBackground(ImageView background) sets a new background |
| public double | getWidth()  returns the double value of width |
| public void | setWidth(double width)  sets the value of width to the new value |
| public double | getHeight()  returns the double value of height |
| public void | setHeight(double height)  sets a new value for the height |

Tanks extends Rectangle implements Serializable

Class Descriptions

The class Tanks extends rectangle because the it acts as the hitbox for the tanks. When the game is saved, this class is written into a file.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private ImageView | texture  The tank’s texture |
| private double | numOfLifePoints  The tank’s amount of life points |
| private Media | movingSound  The sound when the truck moves |
| private MediaPlayer | movingSoundMediaPlayer  The MediaPlayer for the moving sound |
| private MapGeneration | mapGeneration  used to generate the map and find coordinates on it |
| private double | armor  The amount of armor on the tank |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | Tanks()  Default no-arg constructor |
| public | Tanks(ImageView texture, MapGeneration mapGeneration, double rectangleWidth, double rectangleHeight)  The constructor of tanks for non-default values |
| public ImageView | getTexture()  Returns the texture of the tank |
| public void | setTexture(ImageView texture) Sets a new texture for the tank |
| public double | getNumOfLifePoints()  Returns the number of life points |
| public void | setNumOfLifePoints(double numOfLifePoints) Sets a new number of life points |
| public Media | getMovingSound()  Returns the moving sound |
| public void | setMovingSound(Media movingSound) Sets a new moving sound |
| public MediaPlayer | getMovingSoundMediaPlayer()  Returns the media player of the moving sound |
| public void | setMovingSoundMediaPlayer(MediaPlayer movingSoundMediaPlayer) Sets a new media player for the moving sound |
| public MapGeneration | getMapGeneration()  returns the mapGeneration |
| public void | setMapGeneration(MapGeneration mapGeneration)  sets a new MapGeneration for the mapGeneration |
| public double | getArmor()  Returns the amount of armor of the tank |
| public void | setArmor(double armor) Sets a new amount for the tank’s armor |

Player implements Serializable

Class Description

The class player represents the user or the computer. When the AI is activated, the user cannot control this player and the computer plays for it. When the game is saved, this class is written into a file.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private boolean | activateAI  This Boolean serves to decide whether or not this player is a bot |
| private String | playerName  This String stores the player’s name |
| private Tanks | tanks  The tank associated with this player |
| private double | money  This variable stores the amount of cash available to the player |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | Player()  Default no-arg constructor |
| public | Player(String name, boolean activateAI, Tanks tanks, double money)  This constructor sets a tanks, name and money for the player. It also decides if the player is a bot or not |
| public boolean | getActivateAI()  Returns the variable activateAI |
| public void | setActivateAI(boolean activateAI) Sets if the player is controlled by a bot |
| public String | getPlayerName()  Returns the player’s name |
| public void | setPlayerName(String playerName) Sets a new name for the player |
| public Tanks | getTanks()  Returns the tank associated with this player |
| public void | setTanks(Tanks tanks) Sets a new tank for this player |
| public double | getMoney()  Returns the amount of money of this plaeyr |
| public void | setMoney(double money) Sets a new amount of money for this player |

MapGeneration implements Serializable

Class Description

This class generates a map curve with a sine function to create a map with hills. This class also calculates the ballistic trajectory of the weapons and their accurate X and Y position. When the game is saved, this class is written into a file.

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private double | amplitude  Variable to set the amplitude of the wave |
| private double | periodScale  The period of the wave |
| private double | functionHorizontalShift  The shift of the wave on the right or left |
| private double | gravity  The gravity of this map |
| private int | mapType  Sets the type of map to generate |

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| public | MapGeneration()  Default no-arg constructor |
| public | MapGeneration(int mapType, double amplitude, double periodScale, double functionHorizontalShift, double gravity)  Constructor to set the map type, gravity, amplitude, period scale, and horizontal shift |
| public double | functionYValueForX(double xPos)  Returns the y value for the x position |
| public double | travelDistance(double angle, double velocity, double initialYPos) Return the total distance made from start to the end |
| public double | travelTime(double distance, double time, double velocity, double angle)  Returns the time to travel the distance from start to the end |
| public double | yPosForX(double angle, double velocity, double initialYPos, double xPos) Returns the y position for any given x while moving |
| public double | xPosForTime(double initialXPos, double time, double velocity, double angle)  Returns the x position at any given moment |
| public double | getAmplitude()  Returns the amplitude of the wave |
| public void | setAmplitude(double amplitude) Sets a new amplitude for the wave |
| public double | getPeriodScale()  Returns the length of the period of the wave |
| public void | setPeriodScale(double periodScale) Sets a new period scale for the wave |
| public double | getFunctionHorizontalShif()  Returns the horizontal shift of the wave |
| public void | setFunctionHorizontalShif(double functionHorizontalShif) Sets a new horizontal shift for the wave |
| public int | getMapType()  Returns the map type |
| public void | setMapType() Sets a new map type |
| public double | getGravity()  Returns the value of the gravity |
| public void | setGravity(double gravity) Sets a new value for the gravity |

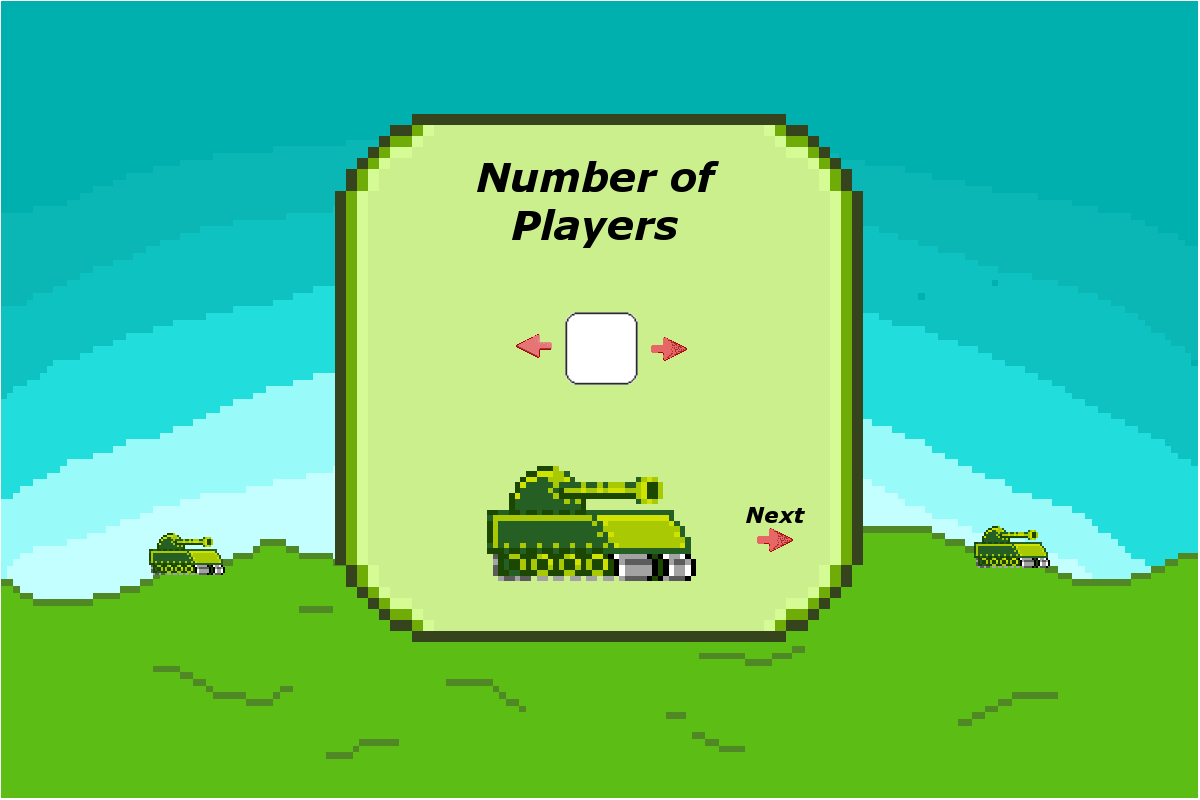
**GUI & Character Design**

Tanks:  
 United States:  
https://lh6.googleusercontent.com/voJwIN2NE4zosVoolw5JaRfbaTY3WCj5vDE2PLswUaSXbqGZT4YZgfD2hVUlvjBr6bk4EgS3xS3ZFEb5Mla373goOi7wdnAl7COx11wgKELLifle81el1irtwYcKiF3VYYEfnWrt  
 Canada:  
https://lh5.googleusercontent.com/Xr_bkefvO37mu40qdZCLYwIPUfVsj1O8nCeDlU3daBmq9ML_vatzSC-UjrxD3MtHolRCzAIGj1C5YNb0jVwtkPvWDKX4Kf9Gj9kUjZk4kmWnW-KCo6WehJfgyBuWa7cn0ItQiy8T  
 North Korea:  
https://lh6.googleusercontent.com/W5k9R1CPRvgE_1o1wISRt9KdF87b4YHC1SZK1TaCQSmK5NiKyFQ4hxeD_lVRF7HUSZU6CBvvWtxdBBepUDuxy0KKEZ2UIJYPo4c8ElTZPtHw1Utcrb0hb37V0t7uzI4hAfiBNARP  
 China:  
https://lh3.googleusercontent.com/tozpm-4IluzWWyOFc5lxWO_A2zXfkd8CxS5CeNLfFyQZUkbXQnfa_INUXsktOAptcqNevcWQZps-xLTANqwfdsSbAX3PTIp4x4vZw0oBrjggVHVZDk12z1Li8yst1l7vbbx2_wED  
  
Items:  
 Weapons:  
 Small:  
 Normal round:  
   
 Shrapnel:  
 https://lh6.googleusercontent.com/S1C200KpFVXynZjnA-rC1c6AtPq2J1LG7_Iy4XYQrRrzCkmi8m85Fz7XwAQz4k40khbd7cAmCuebtcLWZHsvQ0l_uj_vYOL8YZDlfHfUsBLUVHshUo1VnRK8M0beq1cADjNED_Vh  
 Mines:  
 https://lh6.googleusercontent.com/vgFuS_Q4I07lLpXD9FP2EWLY0EcgzABb95PSv9PBkrZcd3xHvrZWAARThfpi7jxBTkRUJXO5OVNU948jB4JV0qQbaD_jTilB_n8i0fBk4P33kBc7qySCU384LkQE9z8n4kVSJKrQ  
 LMG:  
 https://lh5.googleusercontent.com/0OgaK33xDg_FzVcDPsBU3PJPuh3K-24RwShSPp6dRPKbmn5akYDCvU8LPbGFuxGNrbNqEgN0NUZ3HsHoInOSrJn6_ZVyectJYMZh0S4EiE-bVht60KoN0A_kn_QkRujSZYLY0DnJ  
 Medium:  
 Laser:  
 https://lh4.googleusercontent.com/eDlLE1KsKTeerL5y4JDxj9gXlTN0Fu1H1f1kWivNk1fE0IRL2UgA6z_dzD-to3JLRlPNsT-YRHxdwJx-Ihx9VT8nxG0j7NMvPwQk63TPkmVQmzf447vOiiPnNxw7FHegMfT5e8vE  
 C4-RC:  
 https://lh4.googleusercontent.com/Js1MaEFmUOk3zA8FtAlzzBxjkTKrRW7jrKQQYdavMdEGgt7T_hMR4OHcCg5-msnSCPJbRE6wqliLN2MtaJu-1ywAJvl19dMqiuq8rSgVMLu9O5YUQ1SoC_9DS7I2QHyNul02cHHG  
 HMG:  
 https://lh6.googleusercontent.com/Jq7VEY4pzyuyjWGWSwsAtniXMqRHjb-dcVGm5Kv9aDW8lCXH34Jwj_xDdsGjJU5QlqMdM7YYJNBtSdRWeSrwtoI6T7SFffNycx9ePQ6kG5FV8ntlfBxF7kUGr0hzg2SxbGI56TVE  
 Missile:  
 https://lh3.googleusercontent.com/QtOJABo6EeNWW17kxYZ_deGyFzSoUqg8WiJ4aiOY_QZiEO4ZJBII1QGW6IY-wI3uaIr5fJkaNDfoQxbCdfORhBx61--donatTz3lG0AFJlZ2CkEw76oKg_H9X1YfQo8afEzL9ENV  
 Heavy:  
 Air strike:  
 https://lh4.googleusercontent.com/qpU2avG382l1LaVckf1_x1g60tRs3wniuZWonm8IREnc0FqTPq6Ir4KMH5QG8Yht0H91-_CD3LkA5pGWmVDSkMSAofIsi_zt9_uFY6d-aS1IGZ2ZT9_-z8D_qB5Vou3wcJ4SCmN-

Precision strike:  
 https://lh5.googleusercontent.com/4S5O-0mvE469cSuf8s8XSj4upA7ATLoN8FyVIM6uw7ISOWHrxCRFubNC9rbA8xkUaGQ8qVxyIt1ZT184Z5JII5ddcJ1qU8LP667sJMLPedC4KKRPBXZUeNH5TH5hpBoQY6HBTLJr

Atomic bomb:  
 https://lh3.googleusercontent.com/JX-GkpQvDcZ62UeYd3TDs9YfncBk-1z-MrNskGmlUTlNtz0fQHV9ezaPgQ7OX6eyUyeitn3XKLhHL1YNSvcn-KEwlCEQmKhy59_oo8Mf83pGbVdp1zG7WnBE6WSeN7Ta927duyO3  
 Repair Kits:  
 Small repair kit:  
 https://lh6.googleusercontent.com/mQg-R1JD38ni2R2KEcWl9Wvj6skK97EQvE8eNJDWfU3QvRzuyo4yaXZ8f261XmYD3krUgDiS78dr0blzuI4NrFYclyVU0UK4fQVKbB8-HXmUpGTjFw26GZ4TZFJTeBj5_09Z6msl  
 Medium repair kit:  
 https://lh5.googleusercontent.com/Is4O4A1Sl5ZKbCSyhuqRvJ27zlXcqCrLm5HR6hoNQkZE2STKcBrDEElQ0zgGyj8QZY6aDoIwOyhoCWS5gjVY0QkJbuy4YlhquqfxG4IabAs3rGu12-86Q7SMd7e5XBfKBdHgT93f  
 Large repair kit:  
 https://lh4.googleusercontent.com/Azbzp5_b73IXvKq6UOgrIetZHldNr1CLZi7EE5SlpSAuj0vDQ6eH2ploKk2RjoWD6cFSC4-Jat5ZvFYwONksuRxjj55u4qSGRTY3GiDZO-52LlVkmdtfy3Fyq1omUoUycvhl51c6  
 Shields:  
 Small shield:  
 https://lh4.googleusercontent.com/VSYq4p14BxPi3-SCJbRTPROGI6pQ_5BLaxyrbTH0XbmheHDXRLTXSkMlnMM9MGgYfAfuwNKEZlmYNNVHLatCZkVaS5nq1lgAETXrC2epSx1_7aiIF6WWceZdw4at87vvRw0xnVkE  
 Medium shield:  
 https://lh5.googleusercontent.com/Mh947nb5iLeoH9MVrzaepGHRXr6kZNLDdXr5Fdj7wV-i2az-REnJzngxcIPGGqJcOpRH__1yfPfHujnKSk1Qvyq2vdXh_FTyzSYOB2yEYSyLX06yLjeKaA9BorOuW9kv3fDJE4oX

Large shield:  
 https://lh3.googleusercontent.com/vvemzK-n-EaIPM_jFKAoHG09TkmlFBQx67EfK4WnXoyOs3OaMaowRqC-WOdFya4kUbDpRqv8GL4jQpfQGLCvJBbvORVz5I7aHCQAmGklT6PVcKiiVpOPoFPQDGkHzO_13km__QgQ  
Special:  
 Armor upgrade:  
 https://lh4.googleusercontent.com/m9gI8Y_1kJjZcJrA0UWZ4vq7h06Rz8ea9gbBcgThuHMEUSjfeeiPTvoo-ybWD8iJNA-Lz5H3nroMkKeoF_0OtIcBL47RFruTTnoxkn0csBIzHk1w_qStxWhaRvZpOyrOlXddF0Vm   
 Engine upgrade:  
 https://lh6.googleusercontent.com/-6w7BLtAe34gOApIokIUXDwn_QzrWeWbGhMD9jVhcLxsnL5-p0ytvkmRewauwDA3XCDXwAaSvTu4U6XpbG70bn8MCK6hIukp2Mc6e04XwhHtP15KmtqMDJsEskN4MpPj6HAHm35p   
 Teleportation:  
 https://lh5.googleusercontent.com/R4vI4ogn1j8JL3C7AoaJNgh_RcEBm34yHrWPvH4xNuffmhdhkqdiRVA6EtXc4UC95v0gYVT-9WmBmkNu2ocKUBJa_wPQcPhCTWl92RqEOfQP1VEz1NXUpegorB3XYxrm0WQJqROS   
  
Menus:  
 Main menu:  
 

Player selection menu:  
 

Country selection menu:  
   
 Map selection menu:

Pause menu:  
   
 HUD:  
 

**Timeline**

|  |  |  |  |
| --- | --- | --- | --- |
| TASK TO IMPLEMENT | PLAN DATE | ASSIGNED PERSON | NOTES |
| **Design** | **February** 26 | **William & Cedrik** |  |
| **UI Completed** | **March 11** | **Cedrik** |  |
| **Map Generation**  **With Map Styles** | **March 11** | **William** |  |
| **HUD** | **March 18** | **Cedrik** |  |
| **Tanks Moving** | **March 18** | **William** |  |
| **Weapons** | **March 25** | **William** |  |
| **Save / Load** | **April 4** | **Cedrik** |  |
| **Leaderboards** | **April 11** | **Cedrik** |  |
| **Sound** | **April 15** | **William** |  |
| **Physics** | **April 24** | **William** |  |
| **Game Mechanics** | **April 27** | **Cedrik** |  |
| **AI** | **April 30** | **William** |  |
| **Debugging** | **May 4** | **Cedrik & William** |  |

**Tools Used to Create Program**

* IDE: NetBeans 8.2  
  NetBeans 8.2 is the most recent iteration of NetBeans and offers a stable integration of the JavaFX package. This IDE is the most popular for coding GUI in JavaFX. It is for this reason that NetBeans 8.2 is used for this project.
* Platform: Windows 10  
  Windows is the most popular and installed OS across the globe. By making sure that the program runs on Windows 10 computers, the potential clients are far wider. Thus, more people will be able to enjoy the product.
* Packages: JavaFX  
  JavaFX comes pre-packed with Java Development Kit (JDK) and is simple to use. Java allows for a simpler approach to cross-platform coding by offering an intermediate between the OS and the code. These are the two reasons for using JavaFX for this program.
* UML Diagram: Visual Paradigm  
  Visual Paradigm is a full UML maker with a simple and easy to use UI.
* Code storage: GitHub  
  GitHub simplifies the sharing of code by storing it in the cloud and updated when modifications are made. This way, everyone has the same updated version of the code.
* Visual graphics: Gimp   
  Gimp is an easy and free GNU image manipulation program to create the icons and UI elements of the program.

**Features**

Multiplayer

* The game can be played up to 4 players. Local multiplayer meaning that everyone plays on the same computer. Turn based system where everyone plays one after the other. The game cannot be played alone, therefore default value for players is 2, one being a computer.

Can play up to 4 people

* Each player has the choice of 1 of 4 countries to pick from. USA, Canada, North Korea, and China.

AI (player vs computer)

* Players have the option of making another player be controlled by a computer, which allows them to play the game alone.

Character selection

* Each player during the country selection screen can enter their name, and the country they want to fight for. They also have at that moment to activate the AI option for any player.

Possibility to save one game at a time

* One game can be saved at a time if players want to finish what they started. Saving another game will overwrite the current save and erase it.

Air resistance

* Air resistance is implemented to add more difficulty. Projectiles will be affected by it, and resistance will randomly change direction and speed every turn.

Gravity

* Each map has its own gravitational property. However, this property remains the same throughout the game on that map, unlike wind resistance.

4 different maps

* The terrain itself will be randomly generated, but the players have the ability to choose the environment in which they want to play in. Whether it would be in the snow or desert, or maybe in outer space.

Shop to buy items during your turn

* During the players turn, they have the opportunity to go to the shop and purchase different types of weapons and items to help them win. Obviously they need to have enough money.

Money system

* A player will win money by destroying other tanks, winning the round and successfully hitting other tanks. However, they do lose money for using heavy artillery such as air strike, precision strike and atomic bombs as they are very expensive to use. Each player starts with 500$, but making money is a lot harder than losing it so players have to be smart to win.

HUD with following information:

* The HUD is located at the top of the screen and displays everything the player needs to know to make a decision. The HUD updates automatically for every player. It contains the following information:
  + - Main weapon being used
    - Special items available
    - Wind resistance
    - Gravity
    - Movement left
    - Power of the shot

Player movement:

* Each player has a limited movement each turn and most therefore use it strategically. He/she can purchase the engine upgrade to increase the amount of movement available.

Main features of the map:

* Terrain being destructible is a possible feature if there is enough time

Map 1:

* Forest area
* Earth gravity
* Normal variation in terrain (hills)

Map 2:

* Unknown planet
* Lighter gravity
* Floating islands and hills

Map 3:

* Dessert map (post-apocalyptic earth)
* Slightly lighter gravity
* Very flat terrain

Map 4:

* Distant planet
* Very heavy gravity
* Terrain with a lot of hills

**Conclusion**

In conclusion, the development of TANKS will be complexed with the implementation of many features such as saving, loading, map generation, terrain destruction, and realistic ballistic trajectory movements of the weapons. The time given to develop this program (3 months) makes it even more challenging. However, with a strategy and a plan, this will be possible. The main goal of this project is to create a fun and enjoyable game inspired by the video game “Worms”.