Tank-War Game

Project Overview

The term project consists Tank War game. The goal of the projects is to implement the Object oriented concepts and to develop modular code, so that the good portion of game could be reused to implement other games.

Introduction to Tank game:

Tank War game is a two-player game where each player tank fights against each other. Both tanks can move up, down, left, right, rotate and fire bullets. Player tanks has three lives and a health bar. The game ends when any player lose all three of its life. Player1 can be controlled using Up arrow, Down arrow, Left arrow, Right arrow and can fire using Enter key. Player2 can be controlled using W key, A key, S key, D key and can fire using Space key. Three types of powerups are generated randomly at random times and at random locations. Powerups can be collected by Tanks by colliding with it, but if not collected, powerups disappear in few seconds. Game ends when player loses all 3 lives.

Development Environment

A. **Version of Java Used**

1.8.0_161

B. **IDE Used**

Eclipse, Oxygen.3 Release (4.7.3)

Build instructions for the project:

Download the project from the GitHub. Go to IDE Netbeans. Choose File > NewProject then choose Java > Java Project with Existing resources. In the source package, provide the source folder "src" of the local Git repository. Click finish and a project will be created. Go to Run > Clean and build project.

Run instructions for the project:

After successful Clean and Build, go to Run> Set Project Configuration> Customize and provide the main class as com.game.TankGame for Tank game

While running the game on Netbeans, if the game window is larger than the screen size, right click on Netbeans icon on desktop, go to properties --> Compatibility and un-check "Override high DPI scaling behavior" option.

Rules and Controls for Tank game:

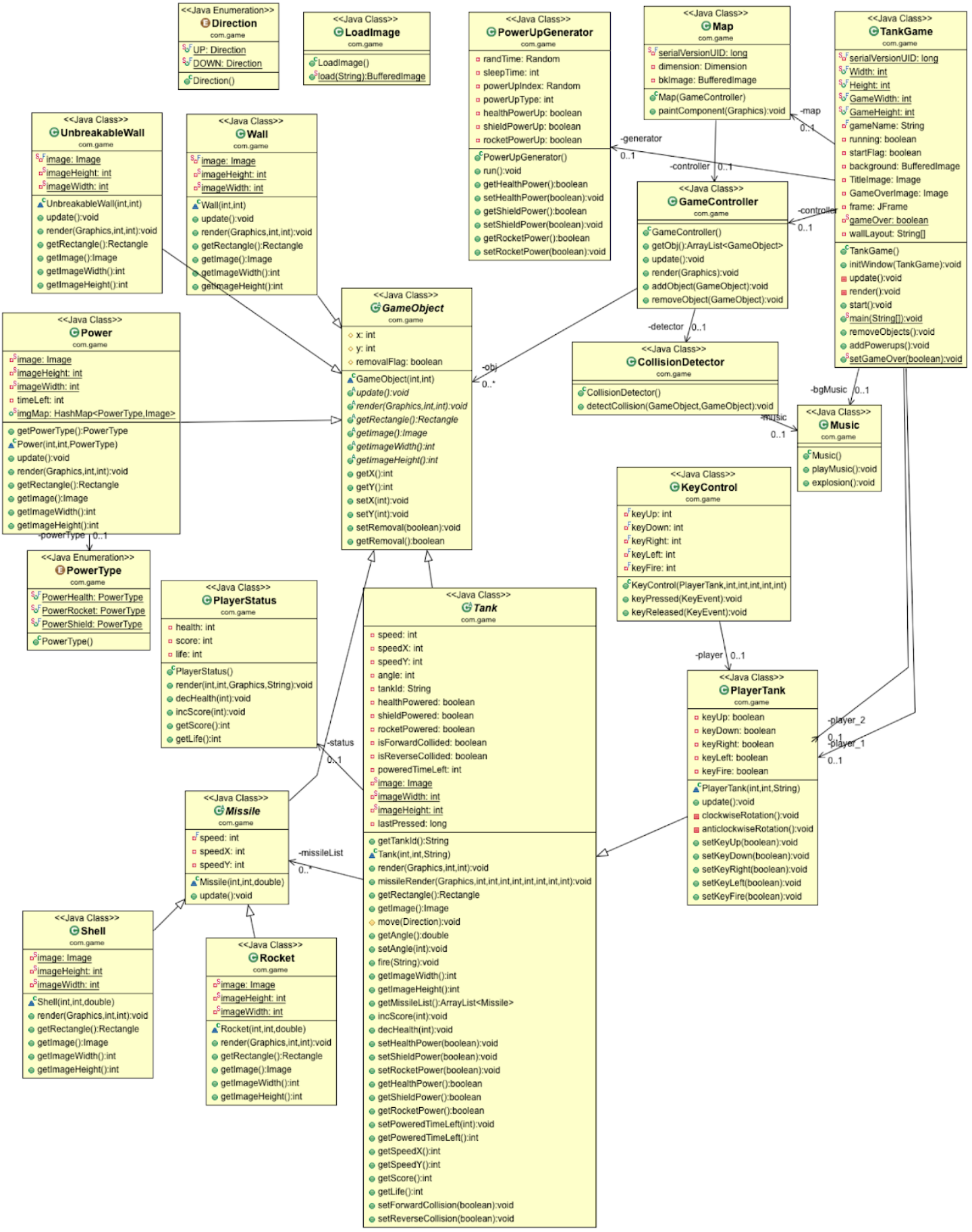
Controls	Player1	Player2
UP	Up Arrow	W
DOWN	Down Arrow	S
LEFT	Left Arrow	A
RIGHT	Right Arrow	D
FIRE	Enter Key	Space key

In the Tank game, Player1 and Player2 fights against each other and can fire missiles. The player tanks cannot pass through breakable and unbreakable walls. However, the breakable wall can be removed by firing.

Assumptions for Tank Game:

- Two player Game.
- Tanks cannot go through the wall.
- No damages can be done to the tanks by colliding with walls.
- Both player tanks have 3 life and a health bar each.
- When health bar gets over, life count decreases by 1. If life count decreases to zero, game ends.
- Upon collecting Health Powerup, health increases by 30%.
- Upon collecting Rocket powerup, extra powerful Missile(Rocket objects) can be fired while shooting for few seconds which cause more damage to enemy tank.
- Upon collecting Shield powerup, the tank is protected against enemy shells and rockets for few seconds.

Tank Game Class Diagram



Class descriptions

GameObject class:

Abstract super class inherited by all game objects like Tank, Missile, BreakableWall, UnBreakableWall, Powerups.

GameObject class in Tank game has following methods.

1. abstract void update() – update method for all game objects.
2. abstract void render(Graphics g , int xOffset, int yOffset) – render method for all game objects.
3. abstract Rectangle getRectangle() – returns the rectangle of game objects.
4. abstract Image getImage() – returns the images of game objects.
5. abstract int getImageWidth() – returns the image width.
6. abstract int getImageHeight() – returns the image height.
7. int getX() – returns the x location of the game object.
8. int getY() – returns the y location of the game object.
9. void setX() – sets the x location of the game object.
10. void setY() – sets the y location of the game object.
11. void setRemoval(boolean val) – sets the removal flag of the game object.
12. boolean getRemoval() – returns the removal flag of the game objects.

CollisionDetector class:

CollisonDetector class is used in Tank game to detect the collision between two game objects and set the respective flags as per the requirements.

CollisionDetector class has following method.

1. public void detectCollision(GameObject o1, GameObject2 o2) - used to check the intersection between rectangles of two GameObject passed as parameters.

In the Tank game the Tank object is passed as first parameter. The intersection is checked between the Tank rectangles and rectangles of BreakableWall, UnBreakableWall, Missile, Power and Tank.

GameController Class:

GameController class is one of the important class in the Tank game. The GameController class contains an arraylist to store all the game objects and has below methods.

1. void update() – The code iterates through the arraylist and call the update method of all gameobjects.

2. void render(Graphics g) – In the render method of the Tank class the spilt screen is implemented. Based on the position of the player tanks on the game window, the co-ordinates are calculated and passed as parameter for render method of all the gameobjects by iterating through the arraylist.
3. void addObject(GameObject) - used to add the game objects into the arraylist.
4. void removeObject(GamObject) - used to remove the game objects from the arraylist.

LoadImage Class:

LoadImage class is used to load all the images in the game. The LoadImage class has the below method.

1. BufferedImage load(String) – The load method takes the filename as the parameter and return the buffered image. If file not found in the resource folder, an exception will be thrown and handled.

KeyControl Class:

KeyControl class is used to define the controls for the game by handling the key events. KeyControl class extends the keyAdapter and overrides the below methods.

1. void keyPressed(keyEvent) - define controls when key is pressed.
2. void keyReleased(keyEvent) – define controls when key is released.

PlayerStatus Class:

PlayerStatus class is used to maintain the score and life and levels in the games and has below methods.

PlayerStatus class methods in Tank game:

1. void render(int, int, Graphics, String) - used to render the health, score and life of playertanks.
2. void incScore(int) – increase the score of the playertank by passed integer value.
3. int getScore() – returns the score of the player.
4. int getScoreLevel1()
5. void setScoreLevel1(int)
6. int getScoreLevel2()
7. void setScoreLevel2(int)
8. void setLevel(int)

Music Class:

Music class is used for playing background music and explosion sound. The Music class has following methods. The sound files are saved in resource folder. The Music class uses the Audio player library.

1. void playMusic() - used for playing background music
2. void explosion() – used for playing explosion sound.

Tank Class:

Tank Class is an abstract class which extends GameObjects and overrides the abstract methods declared in GameObject class. The Tank class will be extended by the PlayerTank class. The Tank class has an arraylist to store the missile game objects so that each player tank has its own set of missiles to fire and TankId for both player Tanks. Each Tank Class has its own following methods.

1. @Override void render(Graphics g, int xOffset, int yOffset) – render the player tanks. Uses the affine transformation to rotate the tank based on the key pressed.
2. String getTankId() : Returns the tank id of the player tank.
3. void move(Direction dir) – Based on the Up or Down direction passed as parameter, the tanks are moved forward or reverse direction. Before updating the tank locations the collision will be checked with flag isFoarwardCollided and isReverseCollided.
4. void missileRender(Graphics g, int x11 , int x12, int y11, int y12, int x21, int x22, int y21, int y22) – Render the missiles of the tanks by iterating through the missile arraylist.
5. double getAngle() – returns the angle of the tank
6. void setAngle(int) – Update the angle of the tank based on passed integer.
7. void fire(String) – fire the type of the missile when space or enter key pressed based on String Shell or Rocket passed as parameter.
8. ArrayList<Missile> getMissileList() – returns the missiles arraylist of the tank.
9. void incScore(int val) – Increase the score by integer val passed as parameter.
10. void decHealth(int val) – decrease the health by integer val passed as parameter.

11. void setHealthPower(boolean val) – Flag is set to indicate health power is collected by the Tank.
12. void setShieldPower(boolean val) - Flag is set to indicate shield power is collected by the Tank.
13. void setRocketPower(boolean val)- Flag is set to indicate rocket power is collected by the Tank.
14. boolean getHealthPower() – returns the flag value of Health power.
15. boolean getShieldPower() - returns the flag value of shield power.
16. boolean getRocketPower() - returns the flag value of rocket power.
17. void setPoweredTimeLeft(int val) – set the timer for powerups collected.
18. int getPoweredTimeLeft() – return the time left for powerups.

PlayerTank Class:

Class for player tanks which extends Tank Class and has below methods.

1. void update() – updates the tank movement based on key input by calling move(), anticlockwiseRotation() and clockwiseRotation() methods of the tank. It also checks the flags set for powerups gained by the player tanks and call the appropriate method based on flag values. The update method of the missiles is called by iterating through the missile arraylist.
2. void clockwiseRotation() – updates the tank angle by +3.
3. void anticlockwiseRotation() – updates the tank angle by -3.
4. void setKeyUp(boolean value) – setter for key up.
5. void setKeyDown(boolean value - setter for key down.
6. void setKeyRight(boolean value) - setter for key right.
7. void setKeyLeft(boolean value) - setter for key left.
8. void setKeyFire(boolean value) - setter for key space/enter.

Missile Class:

Missile class is an abstract class which extends the Gameobject class and overrides the update method to calculate the speedX and speedY. There are two types of missiles could be fired by the player tanks. Shell and Rocket, explained below.

Shell Class:

Shell is a type of Missile fired by the Tank under normal conditions without powerup. Shell class extends the Missile class and overrides the below methods.

1. void render(Graphics g, int xOffset, int yOffset)
2. Rectangle getRectangle()
3. Image getImage()
4. int getImageWidth
5. int getImageHeight()

Rocket Class:

Rocket is a type of missile, fired by the tank upon collecting RocketPowerup. The collected powerup will be active for 1000 counts. After the powerup times out, the tank will fire the missiles of type Shell.
The Rocket class overrides the same methods as Shell class.

UnbreakableWall Class:

Unbreakable Wall is a type of game object and the class extends the GameObject class. As the name suggests, the player tanks cannot break the unbreakable wall. If the Tank collides with the unbreakable wall no damage is done. If the missile collides with the unbreakable wall, missile disappears. UnbreakableWall overrides below methods.

1. void render(Graphics g, int xOffset, int yOffset)
2. Rectangle getRectangle()
3. Image getImage()
4. int getImageWidth
5. int getImageHeight()

Wall Class:

Wall is a type of game object and the class extends the GameObject class. The player tanks cannot go through the wall but can destroy it by firing missiles. If the tank collides with wall, no damage is done. When the player tank destroys the wall by firing bullets its score is increased.
Wall Class overrides the same method as UnbreakableWall class.

Power Class:

Tank game has three types of Powerups health, rocket and shield. These powerups are of type game objects and hence are stored in the main arraylist upon generating. Power class extends the GameObject class. Power class also has an hashmap to store the images of powerups based on powerup type which is stored as a key.
Power class overrides below methods.

1. void update() – based on the time left for powerups, the timer decreases or powerups are removed frpm the arraylist.
2. void render(Graphics g, int xOffset, int yOffset)
3. Rectangle getRectangle()
4. Image getImage()
5. int getImageWidth
6. int getImageHeight()

PowerUPGenerator Class:

PowerUpGenerator generates the 3 types of powerups mentioned above. It implements the Runnable interface to generate the powerups on different thread. In the overridden run() method, PowerUpGenerator generates the random number (powerUpType)between 1,2 and 3 using Random() after every 20 to 40 seconds. If powerUpType is 1 – helath powerup is generated, for 2 – shield powerup is generated and for 3 – rocket powerup is generated.
PowerUpGenerator has the below getter-setter methods.

1. boolean getHealthPower()
2. void setHealthPower(boolean val
3. boolean getShieldPower()
4. void setShieldPower(boolean val)
5. boolean getRocketPower()

6. void setRocketPower(boolean val)

Map Class:

Map class is used for creating minimap on to the game window. Map class exptends the jPanel to draw the game objects using its paint component method. The code iterates through the main arraylist and scale down the images before drawing it onto the panel.

Map class has only one method.

1. void paintComponent(Graphics g)

TankGame Class:

TankGame is the main class where game loop is implemented and programs starts with main(). The TankGame class extends Canvas to draw the game window onto the jFrame. TankGame class also stores the wall layout into an array of strings having U, B and N. Based on the position of these in the array, Wall and Unbreakable wall are place on the game window.

The TankGame class has below methods.

1. void initWindow(TankGame) – The jFrame is initialized and size is set. A start button is created into the jPnale and added into the jFrame. Action listener is implemented to the button. Upon clicking the start button, Map variable and TankGame variable is added to the jFrame.
2. void update() – The update method of the GameController is called.
3. void render() - A graphic component is created using BufferStrategy. The code also checks for gameOver flag in this method and if gameOver flag is set to true, gameover image is displayed and score will be displayed with the winner ID, if not, render method of GameCOntroller is called by passing graphic component.
4. void start() – game loop is implemented in the start() method. PowerUpGenerator is initialized and new thread is starts. For every 60 ticks, the removeObjects() method is called to remove the game objects from the arraylist based on the flag and also powerups are added to the arraylist by calling addPowerups(). After this the update() and render() method is called before the next tick.
5. void removeObjects() – the code iterates through the arraylist and check for the removal flag of the object. If the flag is set to true, object will be removed from arraylist.
6. void addPowerups() – The X and Y co-ordinates are generated randomly and rectangle will be drawn using the (X,Y). The code checks if the generated rectangle intersects with any of the wall objects to make sure that powerups are not generated on top of the wall. Based on the powertype generated in PowerUpGenerator, objects are added to the arraylist.
7. void setGameOver(boolean) – The game over flag is set if life of any of the player tanks becomes zero.

Enum used:

PowerType: PowerHealth, PowerRocket, PowerShield

Direction: UP, Down