Project: Dragoniquarium

5731091221 Poomarin Phloyphisut
5731002421 Korrawe Karunratanakul

1 Game Overview

Dragoniquarium Project is inspired by “Insaniquarium” , the famous Popcap aquarium simulation game. Player’s role in this game is to be controller of the pack of dragons. There’re aliens from other planet try to invade them. Our duties is to protect our pack and get rid of aliens.

1.1 Main menu

This is the first impression of anyone who played this game. Player can start new game by clicking on “New Game” button. (see Figure 1) Player can see overall highscore in “High Score” button. And the game will display highscore panel list top 10 player and their score. (See Figure 2)

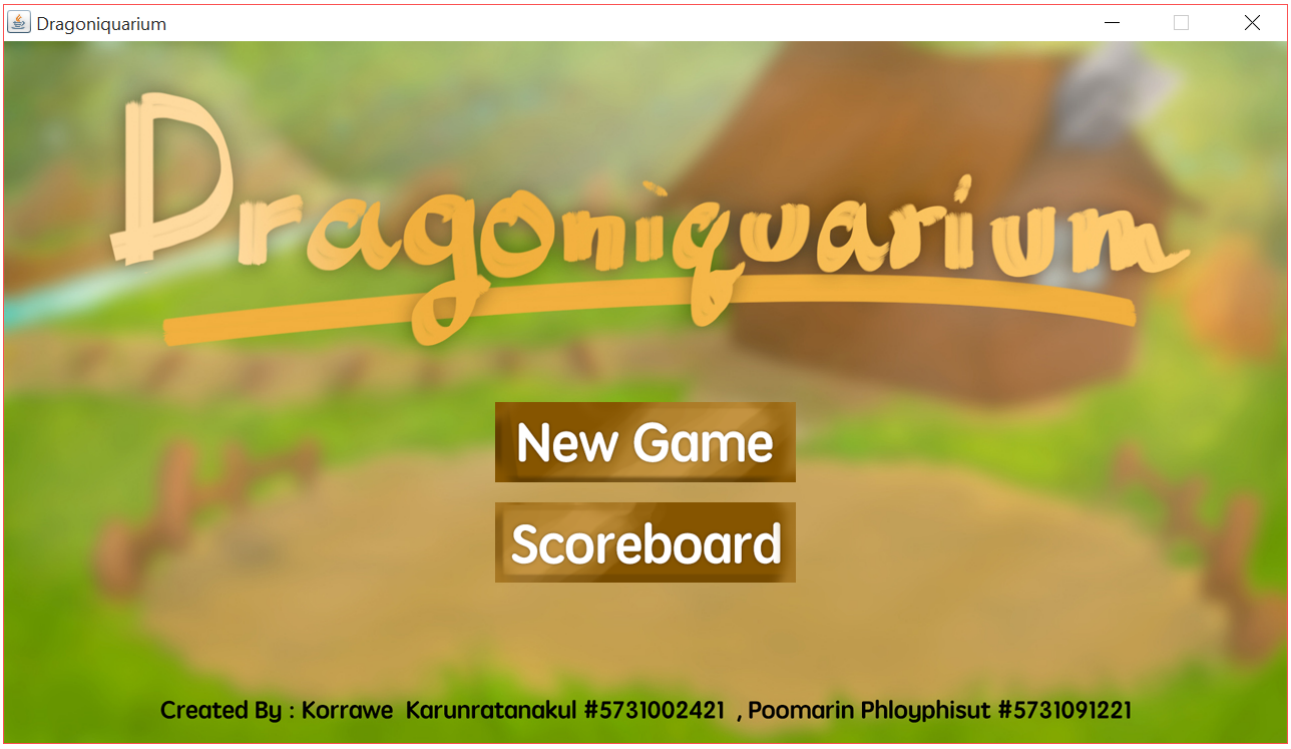


Figure 1: Main menu

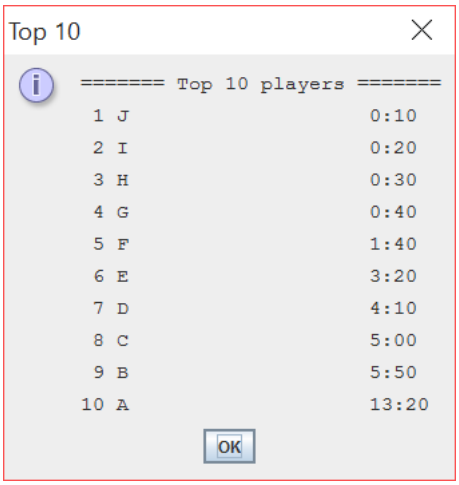


Figure 2 :Highscore Panel

1.2 Main Game

After player press “New Game” button, the scene will transit to this part (See figure 3). In this game player is about to survive 6 wave of enemies and buy golden egg to win.

Figure 3 : Main Game

At any moment, time will be counting and little dragon on above GUI (see figure 4) will be moving to the right. If it reach any flag , the scary alien boss will warp in , then every dragon will perform its work to get rid of the alien. There's 5 type of dragon that player can buy (see table 1).

Figure 4 : Time Line of the game

Type of dragon	Its appearance	Its abilities	Image
Barbigone	green walking dragon. They loves peace and nature. Since they cannot protect themselves so they always need stronger dragon to protect.	laying eggs for collecting to buy other dragon	
Fighter	red flying dragon. These are dragons that are trained to be fighter. they can do basic damage to any invader that is harming them or their friends.	can create fireball to attack alien	
Protector	grey walking dragon. Their main role is protecting Barbigone from being harm. It couldn't attack any thing but yet it has most defence among their friends	can expands the wing to protect Barbigone which is the source of eggs.	

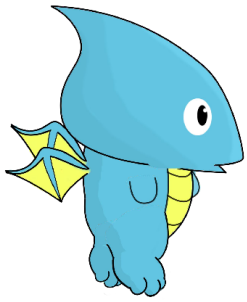
Type of dragon	Its appearance	Its abilities	Image
Petite	blue flying tiny dragon. This is very special dragon that can perform weird ability which is extracting eggs from enemy.	create special plasma ball to extract eggs from alien	
Demon	black gigantic dragon. This is the most powerful dragon that can deal fatally damage to any invader.	create large fireball to attack alien. It will do grave damage.	

Table 1

Player can buy any dragon at any point of the game but if there's no Barbigone (It's all killed) then the player will lose the game. The point of the game is not reaching last boss in timeline but player must buy golden egg as soon as possible. After, last boss there will be no alien left so player will definitely be able to buy golden egg anyway ,but if player wanted to reach top 10 in scoreboard. Player needed to buy golden egg before last boss appear.

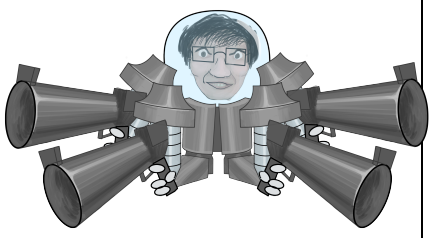
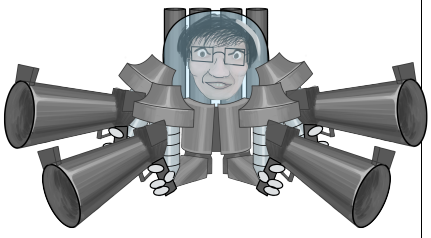
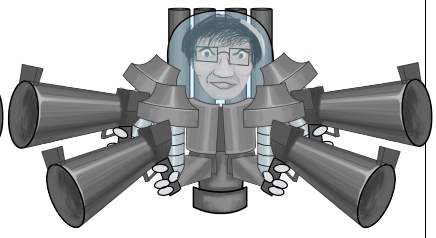
Alien Ozma	Alien Ultima	Alien Omega
		

Table 2

There're 5 type of bosses.(See table 2) The 3 first bosses has same ability which is "Assassinate". This ability is accurately shooting energy ball to player's dragon from afar. The 4rd and 5th alien can perform "Shrapnel" it's the ability of shooting special cannon to the sky then it turn into large amount of energy balls. The 6th(last boss) alien has one more special ability

“Headshot”. It will shoot continuously downward toward our Barbitone it will definitely destroy all dragons beneath.

If player buy golden egg(see figure 5). The game is ended. Overall time that player spends along the game is the score. The less time spent, the more score player get. If player be able to reach top10 player’s high score then game will ask for player name for listing in scoreboard (See figure 6)

2 Implementation Details

This project is divided into 5 packages and one Main class.

2.1 Package “Input”

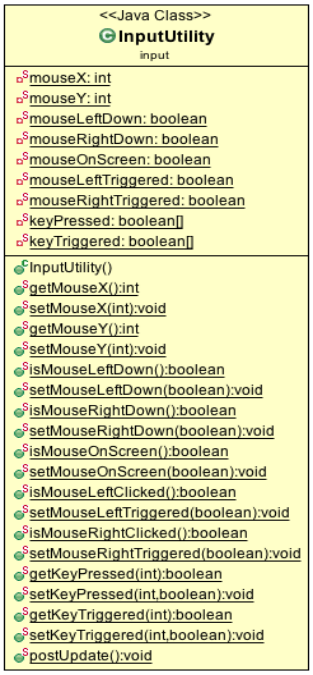
2.2 Package “logic”

2.3 Package “main”

2.4 Package “render”

2.5 Package “ui”

2.1 Package “Input”



UML-1

This package contains all input handler in game.

2.1.1 Class “input.InputUtility”

Field

- int mouseX , mouseY; the position of player’s mouse.
- boolean mouseLeftDown , mouseRightDown; it is true if mouse it pressed.
- boolean mouseOnScreen; It is true if mouse is on screen.
- boolean mouseLeftTriggered , mouseRightTriggered; the trigger variable of mouse button clicking.
- boolean keyPressed[]; the array of all button that is pressed.
- boolean keyTriggered[]; the array of all button that is triggered.

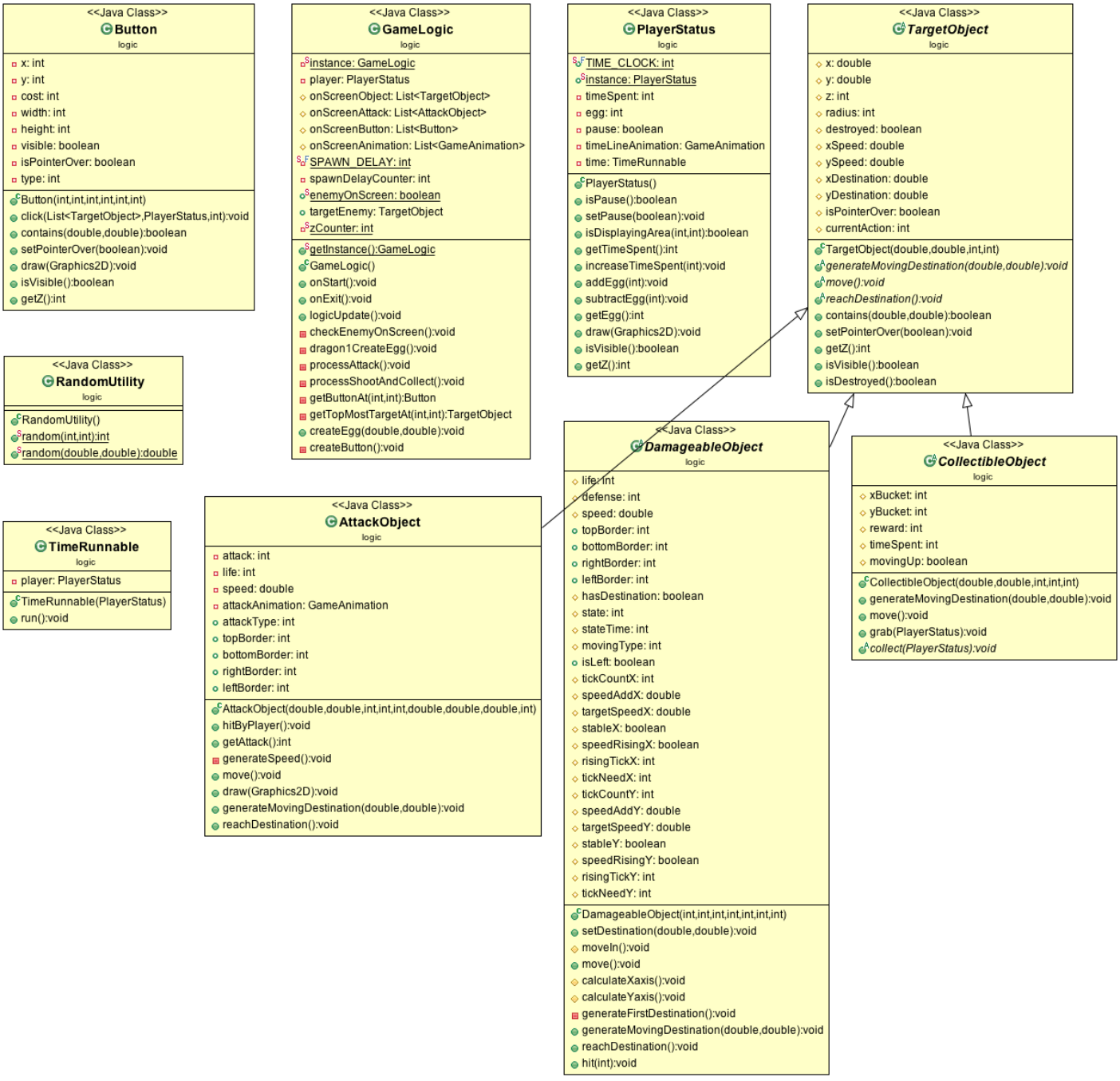
Method

- getters and setters of all variable.
- void postUpdate(); it is called after button is pressed so the game can identify difference between hold and press.

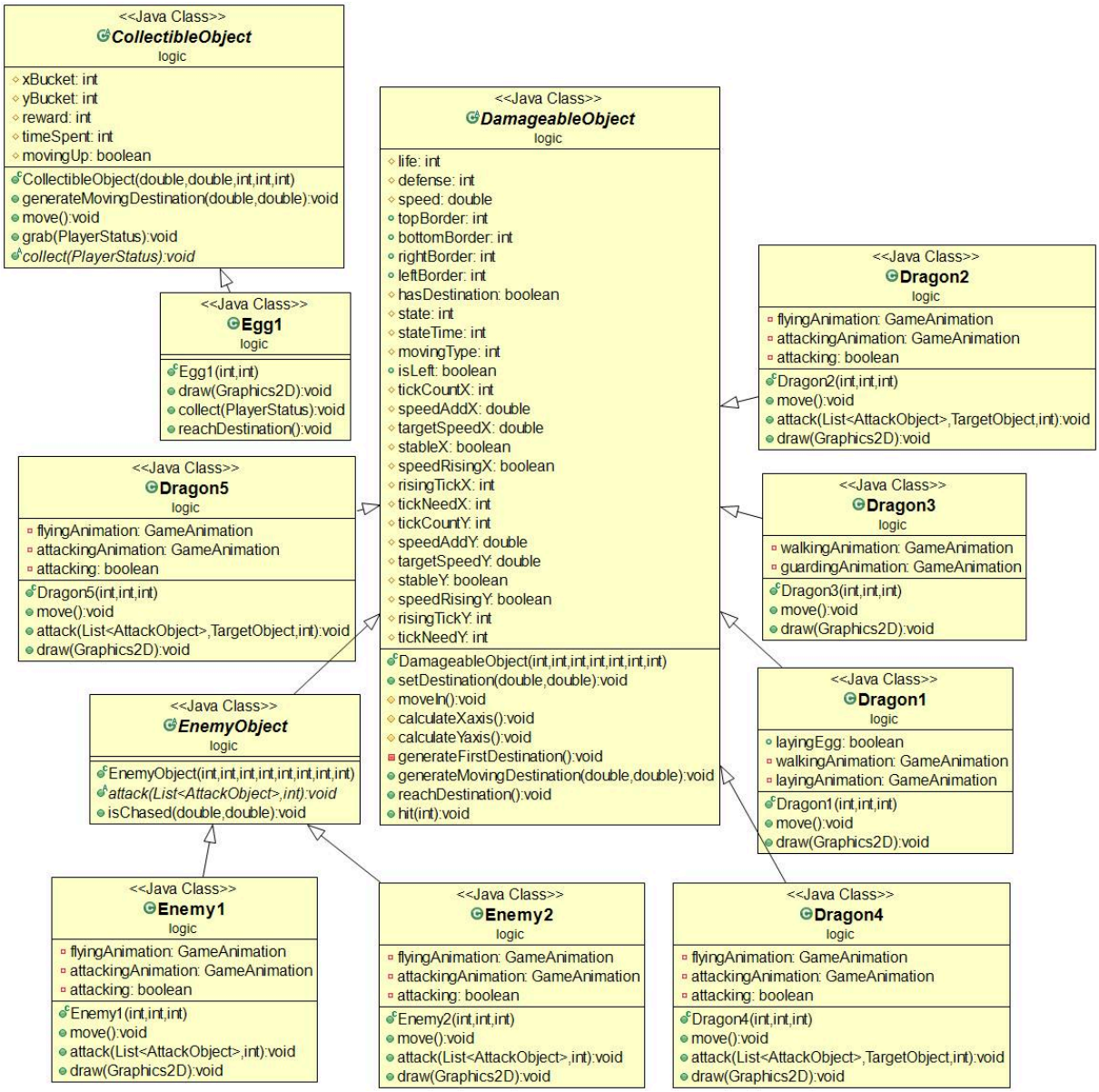
2.2 Package “logic”

This package contains most part of logic in the game.

Since there’re several classes in this package. The UML below (UML 2) is the first half of the package and (UML 3) is the second half.



UML-2



UML-3

3.2.1 Class “logic.AttackObject” extends TargetObject

Field

- int attack; It’s damage this object attacks.
- int life; It’s amount of hit point this object has.
- double speed; It’s speed this object travels.
- GameAnimation attackAnimation; It’s contains attack animation of this object.
- int attackType; It’s type of attacking for this object.
- int topBorder; topmost border it can reach
- int bottomBorder; bottom most border it can reach

- int rightBorder; rightmost border it can reach
- int leftBorder; leftmost border it can reach

Constructor

- AttackObject(double x, double y, int radius, int z, int attack, doublexDes, double yDes, double speed, intattackType); Set every essential variables.

Method

- voidhitByPlayer(); decrease hit point of this object if it's hit point goes zero or negative, it dies.
- intgetAttack(); It's a getter getting amount of attack of this object
- voidmove(); It updates position of next destination to reach bounded by borders.
- voiddraw(Graphics2D g2d); draws this object animation.
- voidgenerateMovingDestination(double curX, double curY); override but not used.
- voidreachDestination(); override but not used.

2.2.2 Class "logic.Button" implements IRenderable

It's class contains every button of main game which is buy button and buy golden egg except for pause button.

Field

- int x, y; It's the position of this button
- int cost; The cost of buying
- int width, height; It's the width and height of this button
- boolean visible; It's true if this button is visible
- boolean isPointerOver; It's true if player's mouse pointer is over this button

type	meaning
1-5	create dragon type 1-5
6	buy golden egg

- int type; it's type of button

Constructor

- Button(int type, int x, int y, int width, intheight,int cost); initialize all essential variable.

Method

- void click(List <TargetObject>onScreenObject, PlayerStatusplayer,intzCounter); this function is called when player click on this button. if this button is buy dragon button and player has enough money, then it will create dragon according to type and if this is buy golden egg button then the game end.
- boolean contains(double x, double y); return true if this button contains point (x,y)
- void setPointerOver(booleanisPointerOver); setter setting isPointerOver variable
- voiddraw(Graphics2D g2d); draw this button by position,width and height.
- boolean isVisible(); return true if this button is visible.
- int getZ();return biggest integer (Integer.MAX_VALUE) because GUI must be on topmost of screen.

2.2.3 Class “logic.CollectibleObject” extends TargetObject

Field

- int reward; number of eggs to be obtained after collect this object
- boolean movingUp; true if this object is moving up and false otherwise.

Constructor

- CollectibleObject(double x, double y, int radius, int z, int reward); initialise all essential variable and set movingUp to true because collectible object must goes up first before falling.

Method

- void generateMovingDestination(double curX, double curY); randomly generate max height of objects before falling.
- void move(); transform x,y toward destination.

- void grab(PlayerStatus player); this is called when player collect this object then destroy this object.

2.2.4 Class “logic.DamageableObject” extends TargetObject

Field

- int life; hit point of this object.
- intdefense; defence to protect this object after being attacked.
- doublespeed; speed this object travels.
- inttopBorder = 20; upmost border this object can reach.
- intbottomBorder = 650; bottommost border this object can reach.
- intrightBorder = 1240; rightmost border this object can reach.
- intleftBorder = 200; leftmost border this object can reach.
- intstate; this is current state of action of this object

state	meaning
1	normal move
2	idle
3	lay egg
4	attack
5	defense

- intmovingType; this is moving type for this object
- boolean isLeft; Assert true if this object facing to the left side
- inttickCountX = 0; Counter to determine acceleration change in X-axis
- double speedAddX = 0; Acceleration in X-axis
- double targetSpeedX = 0;Upperbound of speed in X-axis in the current state
- boolean stableX = false; True if acceleration equals zero
- boolean speedRisingX = false; True if acceleration in X-axis is more than zero

- intrisingTickX = 0; Counter to determine acceleration change in X-axis
- inttickNeedX = 0; Counter for total ticks need to complete each state
- inttickCountY = 0; Counter to determine acceleration change in Y-axis
- double speedAddY = 0; Acceleration in Y-axis
- double targetSpeedY = 0; Upperboundof speed in Y-axis in the current state
- boolean stableY = false; True if acceleration equals zero
- boolean speedRisingY = false; True if acceleration in Y-axis is more than zero
- intrisingTickY = 0; Counter to determine acceleration change in Y-axis
- inttickNeedY = 0; Counter for total ticks need to complete each state

Constructor

- DamageableObject(intx,int y, int radius, int z, intmovingType, int life, int defense);

Method

- void setDestination(double xDes, double yDes);Set destination to xDes, yDes
- void moveIn();Calculate position of thisobject when created
- void calculateXaxis();Calculate X-axis position for the following frame
- void calculateYaxis();CalculateY-axis position for the followingframe
- void generateFirstDestination();Set first destination of thisobject
- voidgenerateMovingDestination(double curX, double curY);
- void reachDestination();Determinenext action when target reach its destination
- void hit(int damage);Decrease life whenthisobjectis hit

2.2.5 Class “logic.Dragon1” extends DamageableObject

This is class for green dragon that lay eggs.

Field

- boolean layingEgg; It will be true if this dragon is laying eggs the default value is false.
- GameAnimationwalkingAnimation; It collects walking animation data.
- GameAnimationlayingAnimation; It collects laying egg animation data.

Constructor

- Dragon1(int x, int y, int z); This function initialise all essential value including x,y,z,stateTime=200,walkingAnimation and layingAnimation.

Method

- void move(); move to destination along x-Axis
- voiddraw(Graphics2D g2d); draw walkingAnimation if this object is no laying egg and draw layingAnimation otherwise.

2.2.6 Class “logic.Dragon2” extends DamageableObject

Field

- GameAnimationflyingAnimation; It collects flying animation data.
- GameAnimationattackingAnimation; It collects attacking animation data.
- intattackTickCount; It count number of time waiting for next attack if it’s attacking
- intattackDelay; It’s delay between attacking if this object is attacking.The default value is 100.
- boolean attacking = false; It’s true if this object is attacking something.

Constructor

- Dragon2(int x, int y, int z); This function initialise all essential value including x,y,z,stateTime=200,flyingAnimation and attackingAnimation.

Method

- void move(); It moves object position
- voidattack(List <AttackObject>onScreenAttack, TargetObjecttargetEnemy, intzCounter); This function is called when enemy object is on screen so this object will keep attacking it and wait by counting until attackDelay.
- voiddraw(Graphics2D g2d); It draws flyingAnimation or attackingAnimation depends on if this object is flying or attacking.

2.2.7 Class “logic.Dragon3” extends DamageableObject

This is a class for grey guarding dragon. This dragon is always walking on ground and be able to protect other dragon from enemy attack.

Field

- GameAnimationwalkingAnimation; It collects walking animation data.
- GameAnimationguardingAnimation; It collects guarding animation data.

Constructor

- Dragon3(int x, int y, int z); This function initialise all essential value including x,y,z,stateTime=200,walkingAnimation and layingAnimation.

Method

- void move(); It transforms this object to other position and other animation state
- voiddraw(Graphics2D g2d); It draws walkingAnimation or guardingAnimation depends on if this object is walking or guarding.

2.2.8 Class “logic.Dragon4” extends DamageableObject

Field

- GameAnimationflyingAnimation; It collects flying animation data.
- GameAnimationattackingAnimation; It collects attacking animation data.
- intattackTickCount; It count number of time waiting for next attack if it’s attacking
- intattackDelay; It’s delay between attacking if this object is attacking.The default value is 100.
- booleanattacking = false; It’s true if this object is attacking something. The default value of this variable is false.

Constructor

- Dragon4(int x, int y, int z); This function initialise all essential value including x,y,z,stateTime=200,walkingAnimation and layingAnimation.

Method

- void move(); It transforms this object to other position and other animation state
- voidattack(List <AttackObject>onScreenAttack, TargetObjecttargetEnemy, intzCounter); This function is called when enemy object is on screen so this object will keep attacking it and wait by counting until attackDelay.

- voiddraw(Graphics2D g2d); It draws flyingAnimation or attackingAnimation depends on if this object is flying or attacking.

2.2.9 Class “logic.Dragon5” extends DamageableObject

Field

- GameAnimationflyingAnimation; It collects flying animation data.
- GameAnimationattackingAnimation; It collects attacking animation data.
- intattackTickCount; It’s counter counting number of time waiting for next attack if it’s attacking
- intattackDelay; It’s delay between attacking if this object is attacking.The default value is 100.
- boolean attacking; It’s true if this object is attacking something. The default value of this variable is false.

Constructor

- Dragon5(int x, int y, int z); This function initialise all essential value including x,y,z,stateTime=200,walkingAnimation and layingAnimation.

Method

- void move(); It transforms this object to other position and other animation state
- voidattack(List <AttackObject>onScreenAttack, TargetObjecttargetEnemy, intzCounter); This function is called when enemy object is on screen so this object will keep attacking it and wait by counting until attackDelay.
- voiddraw(Graphics2D g2d) It draws flyingAnimation or attackingAnimation depends on if this object is flying or attacking.

2.2.10 Class “logic.Egg1” extends CollectibleObject

Constructor

- Egg1(int x, int y); It set all default variable.

Method

- void draw(Graphics2D g2d); It draws image of egg on position (x,y)

- voidcollect(PlayerStatus player); It will be called if player collect egg and player will get some amount of egg as reward.
- voidreachDestination(); this object will be transform position toward destination. It can be either up or down.

2.2.11 Class “logic.EnemyObject” extends DamageableObject

Constructor

- EnemyObject(int x, int y, int radius, int z, intmovingType,int life, int defense, intattackDelay); This will initialise all parent variable including (x , y , radius , z , movingType , life , defense)

Method

- abstract void attack(List <AttackObject>onScreenAttack, intzCounter); this is abstract function to be called when enemy is attacking.
- voidisChased(double xClick, double yClick); This function is called when player click on some part of this object to attack and this object will move away from direction that player attacked. It seems like this object is being chased by player.

2.2.12 Class “logic.GameLogic”

Field

- staticGameLogicinstance; It is a static instance of GameLogic to be used in game
- PlayerStatusplayer; It is current status of player
- List <TargetObject>onScreenObject; It is a list of all TargetObject on screen.
- List <AttackObject>onScreenAttack; It is a list of all AttackObject on screen.
- List <Button>onScreenButton; It is a list of all Button on screen.
- List <GameAnimation>onScreenAnimation; It is a list of all GameAnimation on screen.

- static finalintSPAWN_DELAY = 100; It is static default value of spawning delay of object.
- intspawnDelayCounter; It is counter counting tick until SPAWN_DELAY to spawn some object.
- static boolean enemyOnScreen; It is true if there's at least one enemy on screen at current moment. The default value is false.
- TargetObjecttargetEnemy ;It collects data of enemy.
- staticintzCounter = Integer.MIN_VALUE+1; It is a value of z to assign to new object that is created so there will be no equals value on any object in the game

Method

- staticGameLogicgetInstance(); this return instance of GameLogic
- voidonStart(); this is an initialise function. It set all default value to all variable
- voidonExit(); this will be called when the game is end. It will clear all list of object and instance.
- voidcheckEnemyOnScreen(); this function check if there's any enemy on screen. If there's at least one ,enemyOnScreen will be true.
- voiddragon1CreateEgg(); it will create object "egg" for any green dragon that is entering laying egg state.
- voidprocessAttack(); it will check if any shot hit any object and if the shot must destroy that object. That object will be flagged as destroyed. If the shot will decrease that object hit point. The hit point will be decrease. if the shot must create eggs (blue dragon's shot), new egg must be created at the same position.
- void processShootAndCollect(); this is the main function that is calculate player behaviour from input.InputUtility then it identify if that click is for shooting, collecting or neither.Then all required function will be called depends on type of that click.
- TargetObjectgetTopMostTargetAt(intx,int y); It will return the topmost (lowest value of z) object that contains position (x,y) in it.

- voidcreateEgg(double x, double y); It will create instance of egg at position (x,y) then add to onScreenObject.
- voidcreateButton(); this will create all button needed on screen.

2.2.13 Class “logic.PlayerStatus”

Field

- staticfinalintTIME_CLOCK = 10; It’s default value for time counting.
- staticPlayerStatusinstance; It’s instance of current player’s PlayerStatus.
- inttimeSpent; It’s collect amount of time player spent since player enter Main Game.
- integg; It’s amount of egg player has collected.
- boolean pause; It’s true if current game is paused (pause button is pressed) the default value is false.
- GameAnimationtimeLineAnimation; It’s an animation displaying current timeline and current location on timeline.
- TimeRunnabletime; for collecting amount of time player spent through the entire game.

Constructor

- PlayerStatus(); this initialise all variable including egg,timeSpent,timeLineAnimation and Thread.

Method

- boolean isPause(); getter getting pause.
- voidsetPause(boolean pause); setter setting pause.
- intgetTimeSpent(); getter getting timeSpent.
- voidincreaseTimeSpent(int time); increase timeSpent by time.
- void addEgg(int a); increase egg by a
- void subtractEgg(int a); subtract egg by a but it will never goes below 0
- intgetEgg(); getter getting egg.
- voiddraw(Graphics2D g2d); draw overall GUI at the right position including button(graphical) and timeLineAnimation.

- boolean isVisible(); this always return true since this is GUI.
- intgetZ(); this return max int available since this is GUI.

2.2.14 Class “logic.RandomUtilitiy”

Method

- staticintrandom(intstart, int end); return random integer in range [start,end].
- static double random(double start, double end); return random double in range [start,end].

2.2.15 Class “logic.TargetObject” implements IRenderable

Field

- doublex,y; position of this object.
- intz; position of this layer (maximum value is on topmost).
- intradius; radius of this object
- boolean destroyed; It’s true if this object is destroyed. The default value is false.
- doublexSpeed, ySpeed = 0; It’s the speed of this object.
- doublexDestination,yDestination; It’s current destination (x,y) of this object.
- boolean isPointerOver = false; It’s true if player’s mouse pointer is over this object.
- intcurrentAction; the current action that this object is performing.

Constructor

- TargetObject(double x, double y, int radius, int z); This initialise all variable including x, y, radius and z.

Method

- public abstract void generateMovingDestination(double curX, double curY); abstract function to generate destination.

- public abstract void move(); abstract function for transforming position toward destination.
- public abstract void reachDestination(); abstract function for reaching position of destination.
- boolean contains(double x, double y); return true if position (x,y) in inside current object.
- voidsetPointerOver(booleanisPointerOver); setter setting isPointerOver.
- intgetZ(); return current z.
- boolean isVisible(); return true if this object is not destroyed.
- boolean isDestroyed(); getter getting destroyed.

2.2.16 Class “logic.TimeRunnable” implements Runnable

Field

- PlayerStatusplayer; current player set current time counts.

Constructor

- TimeRunnable(PlayerStatus player); set player.

Method

- voidrun(); this will do infinite loop and keep counting time and will be terminated if this thread is terminated.

2.2.17 Class “logic.Enemy1” extends EnemyObject

Field

- GameAnimation flyingAnimation; It collects animation sprite of flying enemy.
- GaneAnimation attackingAnimation; It collects animation sprite of attacking enemy.

Constructor

- Enemy1(int x,int y,int z); initialize all essential variable and set stateTime to 30;

Method

- void move(); this will transform this object position toward destination and also update its state.
- void attack(List <AttackObject> onScreenAttack,int zCounter); this perform enemy's attacking motion including play animation and creating plasma balls.
- void draw(Graphics2D g2d); this perform drawing enemy sprite either flying or attacking.

2.2.18 Class "logic.Enemy2" extends EnemyObject

Field

- GameAnimation flyingAnimation; It collects animation sprite of flying enemy.
- GaneAnimation attackingAnimation; It collects animation sprite of attacking enemy.

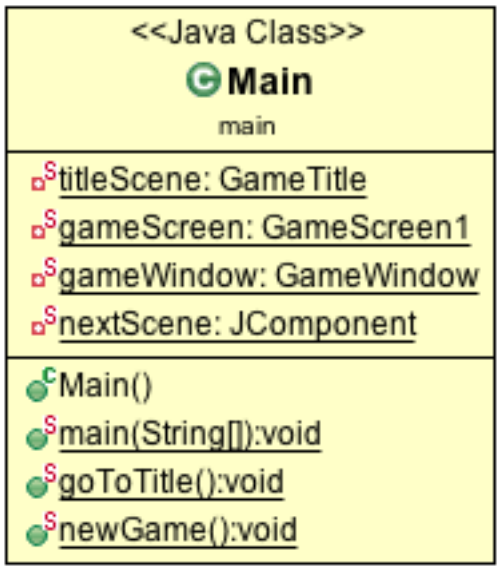
Constructor

- Enemy2(int x,int y,int z); initialize all essential variable and set stateTime to 30;

Method

- void move(); this will transform this object position toward destination and also update its state.
- void attack(List <AttackObject> onScreenAttack,int zCounter); this perform enemy's attacking motion including play animation and creating plasma balls. It plasma ball is different to Enemy1.
- void draw(Graphics2D g2d); this perform drawing enemy sprite either flying or attacking.

2.3 package main



This is the package contain main class of the whole game

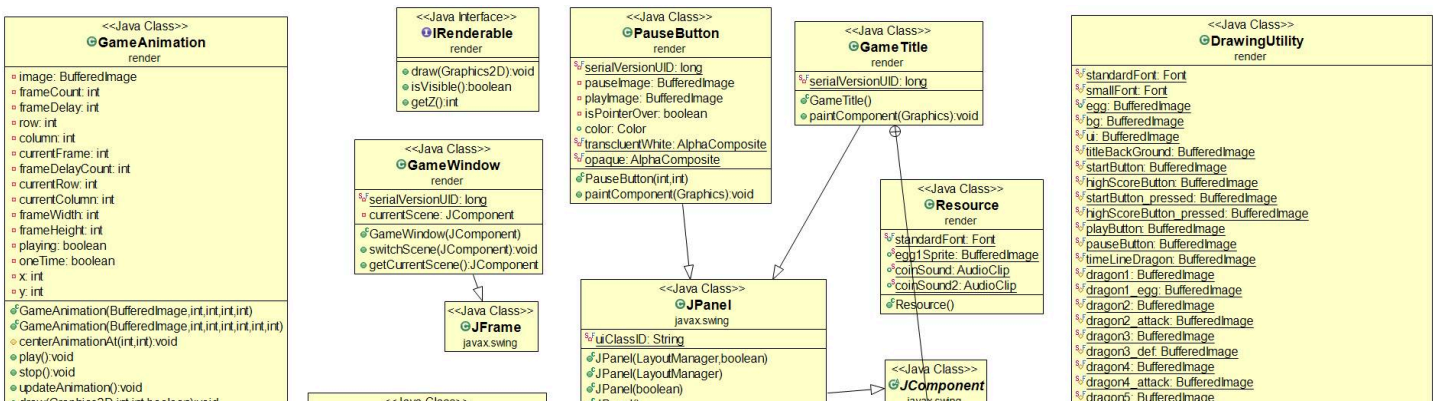
2.3.1 Class “main.Main”

Method

- static void main(String[] args); this will initialise every variable then it will goes infinite loop but sleep for 20ms between each loop and repaint current scene every loop.
- static void goToTitle(); set nextScene to titleScene so scene will goes back title.
- static void newGame(); set nextScene to gameScene so game will start new game.

2.4 package render

This is the package contains all rendering of objects and animations.



2.4.1 Class “render.DrawingUtility”

Field : contain every essential data for drawing everything on screen

Method

- staticBufferedImagegetImage(String directory); get image from specific location
- BufferedImagegetButton(String name); return image of specific button which is “start” , “high score” , “play” and “pause”.
- staticGameAnimationcreateTimeLineAnimation(); create new game Animation for timeLine
- staticGameAnimationcreateAttackAnimation(); create new game Animation for timeLine
- staticGameAnimationcreateDragon1Animation(); create new game Animation for Dragon 1
- staticGameAnimationcreateDragon2Animation(); create new game Animation for Dragon 2
- staticGameAnimationcreateDragon3Animation(); create new game Animation for Dragon 3
- staticGameAnimationcreateDragon4Animation(); create new game Animation for Dragon 4
- staticGameAnimationcreateDragon5Animation(); create new game Animation for Dragon 5
- staticGameAnimationcreateDragon1AnimationLayingEgg(); create new laying egg Animation for Dragon 1
- staticGameAnimationcreateDragon2AnimationAttack(); create new attack Animation for Dragon 2
- staticGameAnimationcreateDragon3AnimationDef(); create new defence Animation for Dragon 3

- staticGameAnimationcreateDragon4AnimationAttack(); create new attack Animation for Dragon 4
- staticGameAnimationcreateDragon5AnimationAttack(); create new attack Animation for Dragon 5

2.4.2 Class “render.GameAnimation”

Field

- BufferedImageimage = null; It’s the sprite of the animation.
- intframeCount,frameDelay; count number of tick by frameCount until frameDelay to advance to next frame then frameCount goes back to 0
- introw,column; number of row and column of image in sprite.
- intcurrentFrame,frameDelayCount; Current frame of animation. and delay count of animation.
- intcurrentRow, currentColumn; Current position of row and column in sprite.
- intframeWidth,frameHeight; Width and height of sprite
- boolean playing; It’s true if current animation is playing. The default value is false.

Constructor

- GameAnimation(BufferedImageimage,intframeCount, int row, intcolumn,intframeDelay); Set every variable to specific value. If the image is not found every value will be set to 0.

Method

- void play(); set currentFrame,currentRow,currentColumn to 0 and playing to true
- void stop(); set currentFrame,currentRow,currentColumn to 0 and playing to false
- voidupdateAnimation(); increase frameCount, if frameCount equal frameDelaycurrentFrame increase then frameCount reset to 0.
- void draw(Graphics2D g2, int x, int y, booleanreflex); draw current animation at specific position(x,y) of the sprite
- Getter and Setter of all variable

2.4.3 Class “render.GameScreen1” extends JComponent

Constructor

- GameScreen1(); set every essential value for frame and set size to 1280x700 and add listener.

Method

- voidaddListener(); Add all mouse listener and handle all event via input.InputUtility's function.

2.4.4 Class "render.GameTitle" extends JPanel

This is the class of main menu scene extends from JPanel. It contains subclass for title button ("New Game" and "High Score").

Field

- subclass TitleButton extends JPanel; this class is for "New Game" and "High Score" button

Field of subclass

- BufferedImageimage; It contains image of the button
- boolean isPointerOver = false; It'll be true if player's mouse is over this button

Constructor of subclass

- TitleButton(String name, int x, int y); Initialise image and set bound of this button

Method of subclass

- voidpaintComponent(Graphics g); paint this button image.

Constructor

- GameTitle() initial all size and buttons.

Method

- paintComponent(Graphics g) paint everything in this scene.

2.4.5 Class "render.GameWindow" extends JFrame

This class extends Frame. It's for setting every essential value for frame of the whole game

Field

- JComponentcurrentScene; It collect the current scene player is currently at.

Constructor

- GameWindow(JComponent scene); It set all essential value for frame and set current scene to scene. This also load new cursor for player's mouse.

Method

- voidswitchScene(JComponent scene); It set current scene to new scene.
- JComponentgetCurrentScene(); Getter getting current scene.

2.4.6interfaceIRenderable

Method

- void draw (Graphics2D g2d); for drawing graphic 2d
- boolean isVisible(); Getter getting visibility of this object
- intgetZ(); Getter getting z value for showing current layer of object

2.4.7 Class "render.RenderableHolder"

Method

- voidadd(GameAnimationanim); This method add and also sort all Renderable object by value z.

2.4.8 Class "render.Resource"

This class is for collecting all data needed for running the game.

Field

- static final Font standardFont = new Font("Tahoma", Font.BOLD, 30);
- staticBufferedImageegg1Sprite;
- static AudioClip coinSound;
- static AudioClip coinSound2;

2.4.9 Class "render.PauseButton" extends JPanel

This is the class handle pause button of the game. It use wait() and notifyAll() on other Threads to pause and continue the game.

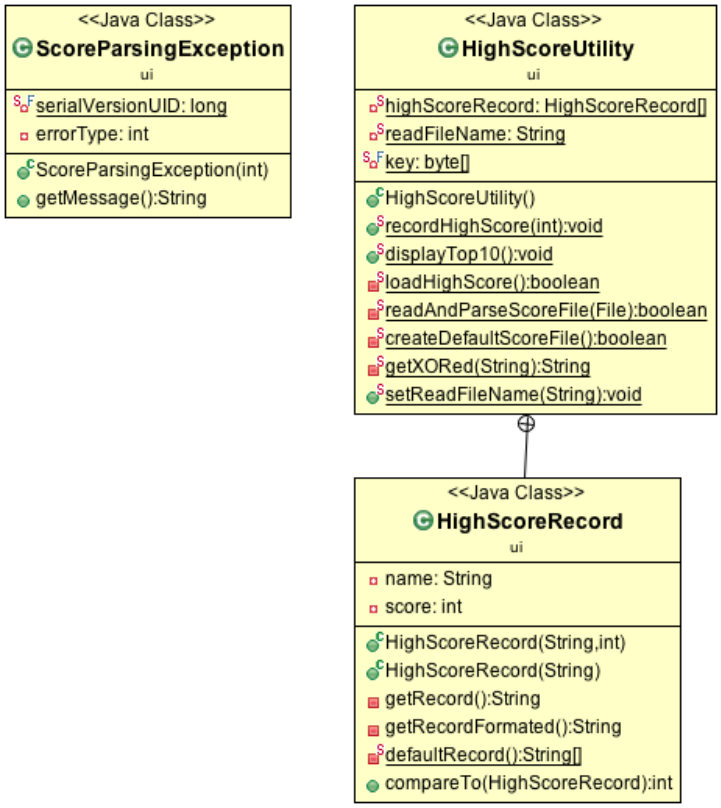
Field

- BufferedImagepauseImage; It contains image of pause button.
- BufferedImageplayImage; It contains image of continue button.

- boolean isPointerOver = false; This will be true if player's mouse pointer is over this button

Constructor

- PauseButton(int x, int y); This initialise all value and also add mouse listener for handling mouse event.



Method

- voidpaintComponent(Graphics g); It paints image of button.

String getMessage(); if errorType is 0 then it means there's no recorded score. If errorType is 1 then is means the format of file is wrong.

2.5 package ui

This is the package contains all about player's scoreboard and highscore.

This is the package contains all about player's scoreboard and highscore.

2.5.1 Class "render.HighScoreUtility"

Field

- staticHighScoreRecord[] highScoreRecord = null;
- static final byte[] key = "RmAAq2b5d8fjgu9dhher".getBytes(); key to be XOR (encrypt) the highscore data in case of being hacked.
- static String readFileName = "highscore"
- SubclassHighScoreRecord implements Comparable<HighScoreRecord>

Field of Subclass

- String name; It's name of player who got highscore
- intscore; number of score he/she got

Constructor of Subclass

- HighScoreRecord(String record); initialise name and score the score need to be parse from string which is time based.

Method of Subclass

- String getRecord(); Getter getting record
- String getRecordFormated(); Getter getting formatted record to display.
- intcompareTo(HighScoreRecord o) comparing highscore for sorting

Method

- static void recordHighScore(int score); record player's highscore after game ended also get name of that player.
- static void displayTop10(); this create JOptionPane for showing the top10 best player (least amount of time).
- staticbooleanloadHighScore(); load highscore file.
- static String getXORed(String in); get data XORed with key;

2.5.2 Class "render.HighScoreUtility"

Field

- int errorType; type of error occured

Method

- String getMessage(); if errorType is 0 then it means there's no recorded score. If errorType is 1 then is means the format of file is wrong.