PROJECT : SPACE CONQUEROR X

By

Nat Srireunthong 5731065021

Nuttapat Kirawittaya 5731035121

2110215 Programming Methodology

Chulalongkorn University

How to Play (For Default Key)

Use Left Arrow Key to move “Left”

Use Right Arrow Key to move “Right”

Use Z Key to Fire “Normal Bullet”

Use X Key to Fire “Piecing Bullet’

Use C Key to Fire “Icy Bullet”

Use P for “Pause Game” or “Resume Game”

Player also can config any key in “Option Panel”

This is a normal enemy.

Enemy can also shoot a bullet.

This is a BOSS!

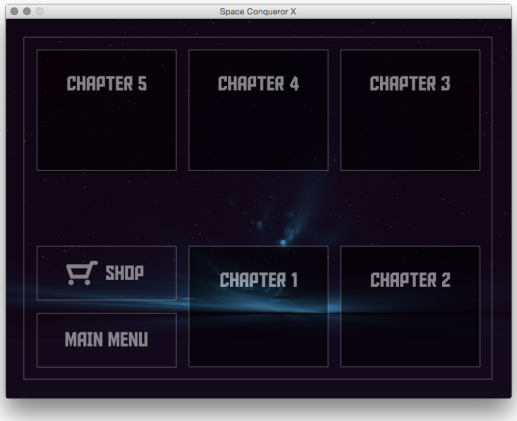
Boss also can shoot a bullet.

This is ‘‘Main Menu’’

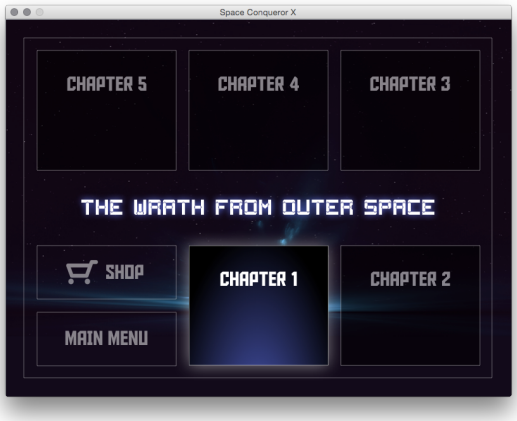
There is ‘‘Help Panel’’ for How-To-Play this game

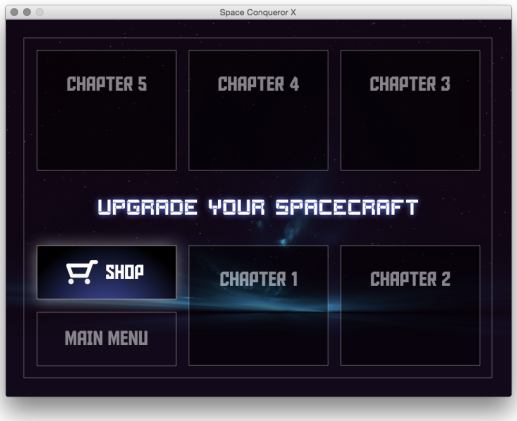
There is also ‘‘Option Panel’’ for config key and mute sound

After you pressed button “PLAY” you will enter “Chapter Select”

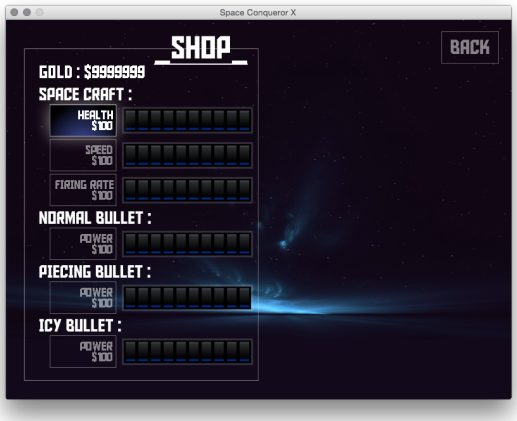


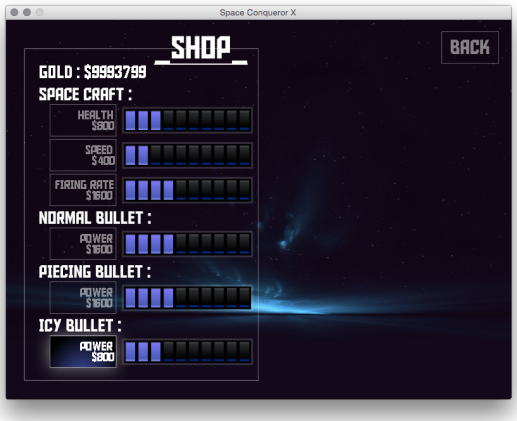
You can choose stage you want to play and also select shop for upgrading your spacecraft



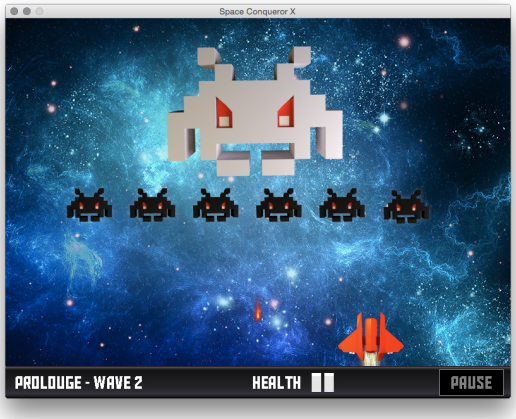


In shop there are six parameters that can be upgraded





This is when you enter a stage



If you win, you will loot some gold

And if you lose, you will get penalty of losing gold and your health will reset to one

Project : SpaceConquerorX

About this project:

Our project is a single player role-playing game. The objective of the game is to win the last boss. Player need to pass all five chapters which including five stage each to win the game. After complete any stage, the next stage will unlock and player will gain some gold so that he or she can upgrades his own spacecraft. The spacecraft got three types of bullet with the common firing rate, player can upgrade any kind of bullet's power, spacecraft's movement speed and also firing rate. Player also can buy some health before enter the stage too. If player lose or quit t a stage, everything he or she earn in that will not be count. This game also can be saved too.

Class “utility.ImageSpriteLoader”

this class is for loading all images for this project

Field

+ static Polygon[][] SpacecraftCollision; collision box for spacecraft (for checking if this collide with others or not)
+ static Polygon[][] BulletCollision; collision box for bullet (for checking if this collide with others or not)
+ static Polygon[][] OneEnemyCollision; collision box for one enemy (for checking if this collide with others or not)
+ static BufferedImage[][] SpacecraftSprite; array of images for spacecraft in every frame
+ static BufferedImage[][] BulletSprite; array of images for bullet in every frame
+ static BufferedImage[][] TitleScene; array of images for TitleScene
+ static BufferedImage[] Completed; images for AfterStageScene
+ static BufferedImage[] Failed; images for AfterStageScene
+ static BufferedImage Play1; a button for TitleScene
+ static BufferedImage Play2; a button for TitleScene
+ static BufferedImage Exit1; a button for TitleScene
+ static BufferedImage Exit2; a button for TitleScene
+ static BufferedImage Help1; a button for TitleScene
+ static BufferedImage Help2; a button for TitleScene
+ static BufferedImage Option1; a button for TitleScene
+ static BufferedImage Option2; a button for TitleScene
+ static BufferedImage CloseButton1; a button for Help, Option
+ static BufferedImage CloseButton2; a button for Help, Option
+ static BufferedImage LV_MAX; a button for Shop
+ static BufferedImage Levelled; a gauge image for Shop

+ static BufferedImage PlayerStatus; a image of bar for PlayerStatus

Methods

+ static BufferedImage[][] loadTitleScene(); return array of images for TitleScene
+ static BufferedImage loadSingleImage(String s); return an image from directory s
+ static BufferedImage[][] load(String s, int w, int h); Return array of Image that loaded directory s and cut it in to w * h size
- static Polygon[][] generateSpacecraftBox(BufferedImage[][] img); for generate collision box for spacecraft
- static Polygon[][] generateBulletBox(); for generate collision box for bullet
- static Polygon[][] generateOneEnemy(); for generate collision box for one enemy

Class "utility.InputUtility"

this class is for all input on this JFrame

Field

- static int mouseX; Position of mouse in x-axis
- static int mouseY; Position of mouse in y-axis
- static boolean mouseLeftDown; true when press left click
- static boolean mouseRightDown; true when press right click
- static boolean mouseOnScreen; true when mouse is on screen
- static boolean mouseLeftTriggered; true when press and release left click
- static boolean mouseRightTriggered; true when press and release right click
- static boolean[] keyPressed; boolean for which keys are being pressed

- static boolean[] keyTriggered; boolean for which keys are triggered
+ static int[] keyBinding; array of keyCode that will use in game
+ static final int LEFT; index of array keyBinding for moving left
+ static final int RIGHT; index of array keyBinding for moving right
+ static final int NORMAL; index of array keyBinding for shooting normal bullet
+ static final int PIECE; index of array keyBinding for shooting piecing bullet
+ static final int ICE; index of array keyBinding for shooting icy bullet
+ static final int PAUSE; index of array keyBinding for pausing the game

Getters / Setters

+ static int getMouseX(); return mouseX
+ static void setMouseX(int mouseX); set value of mouseX
+ static int getMouseY(); return mouseY-22 because JFrame took about 22 pixels for top bar
+ static void setMouseY(int mouseY); set value of mouseY
+ static boolean isMouseLeftDown(); return mouseLeftDown
+ static void setMouseLeftDown(boolean mouseLeftDown); set mouseLeftDown
+ static boolean isMouseRightDown(); return mouseRightDown
+ static void setMouseRightDown(); set mouseRightDown
+ static boolean isMouseOnScreen(); return mouseOnScreen
+ static void setMouseOnScreen(boolean mouseOnScreen); set mouseOnScreen
+ static boolean isMouseLeftClicked(); return mouseLeftTriggered
+ static void setMouseLeftTriggered(boolean v); set mouseLeftTrigged
+ static boolean isMouseRightClicked(); return mouseRightTriggered
+ static void setMouseRightTriggered(boolean v); set mouseRightTriggered

+ static boolean getKeyPressed(int key); get boolean of keyPressed[] at index key
+ static void setKeyPressed(int key, boolean pressed); set boolean of KeyPressed[] at index key
+ static boolean getKeyTriggered(int key); return boolean of keyTriggered[] at index key
+ static void setKeyTriggered(int key, boolean pressed); set boolean of keyTriggered[] at index key
+ static void setKeyBind(int index, int key); set keyBinding at index index by key
+ static int getKeyBind(int index); get keyCode at index index
+ static boolean isLeft(); return true if left is pressed
+ static boolean isRight(); return true if right is pressed
+ static boolean isFireNormal(); return true if shooting normal bullet pressed
+ static boolean isFirePiece(); return true if shooting piecing bullet pressed
+ static boolean isFireIce(); return true if shooting icy bullet pressed
+ static boolean isPauseButton(); return true if pause button pressed
+ static void postUpdate(); update to reset triggered things

Class "utility.PlayerConstant"

this class is for collecting most of parameters for player in this game

Field

+ static final double MAX_HEALTH; maximum hp of spacecraft
+ static final int MAX_SPEED; maximum speed of spacecraft
+ static final double MAX_DAMAGE; maximum damage of each type of bullet
+ static int HEALTH; current health

+ static double DAMAGE; current damage for normal bullet

+ static double DAMAGE_PIECE; current damage for piecing bullet

+ static double DAMAGE_ICE; current damage for icy bullet

+ static double SLOW_ICE; current slow rate for icy bullet

+ static double SPEED; current speed of spacecraft

+ static int FIRERATE; current firerate of spacecraft

+ static int GOLD; current gold

+ static int UPGRADE_LEVEL; current level of each

+ static int SLOW_DURATION; slow duration of icy bullet

- static final byte[] key; code for xor with string

Method

+ static void decreaseHealth(); decrease health

+ static boolean loadPlayerStatus(); load player status true if loadable

+ static boolean createDefaultPlayerStatusFile(); create save file

+ static boolean savePlayerStatusFile(); save game

- static static String getXORed(String in); encrypt or decrypt save file

- static boolean readAndParseFile; load save file or create save file if unable to load

Class "utility.GameLoader"

this class is for loading game

Field

+ static int numberWave; number of wave in that stage

+ static WaveLoader[] wave; information of all waves

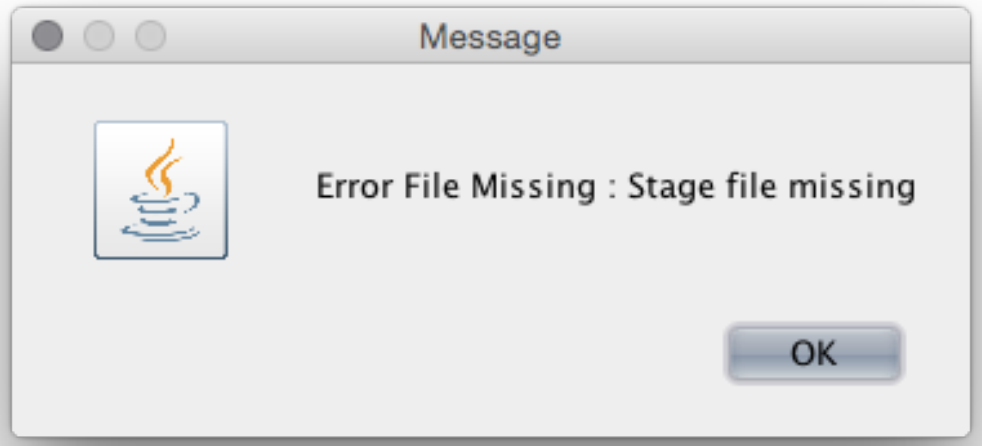
+ static String StageName; name of stage

+ static int Stage; Chapter number

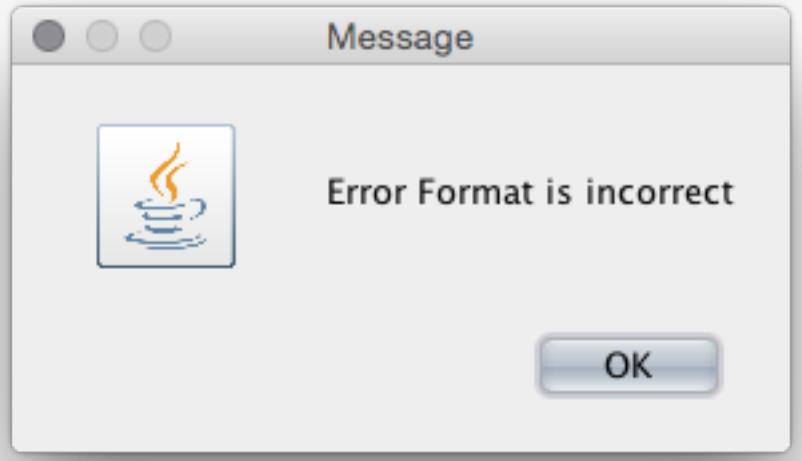
+ static int SubStage; Stage number

Method

+ static void loadStage(int stage, int substage) throws FileMissingException, FileFormatException; load each stage and will throw exception when error occurs, if file is missing this window will poop up



and if file format is incorrect, this window will also pop up



+ static boolean updateKeyBind(); update and save KeyBinding

Class "utility.WaveLoader"

this class is for loading each wave information

Field

- String waveName; names of wave

- CopyOnWriteArrayList<EnemyGroup> enemyGroups; list of enemyGroups

Constructor

+ WaveLoader(String str) throws FileFormatException

Getters

+ getWaveName();

+ getEnemyGroup();

Class "utility.AudioUtility"

this class if for loading all audio in this game

Field

- static boolean MUTED; true when game is muted
- static AudioClip acShoot; sound when enemy is shotted
- static AudioClip acExplosion; sound for explosion
- static AudioClip acClick; sound for click
- static AudioClip acLaserSound; sound for shoot
- static AudioClip acThemeSong; theme music

Getter

+ static boolean isMUTED();

Method

+ static AudioClip loadSound(String s); load file sound at directory s
+ static void playSound(String identifier); play sound
+ static void toggleMUTED(); toggle boolean MUTED

Class "game.gamescene.AfterStageScene" extends JComponent implements GameScene

this class is for creating scene after player completed or failed each stage

Field

- static AlphaComposite halpha; alphacomposite for drawing health bar

- static Font font; font for this screen
- BufferedImage[] ending; image for each state
- int GoldLoot; loot from that stage
- int state; state that will tell which picture will be shown
- int x; position of mouse in x-axis
- int y; position of mouse in y-axis
- boolean Pass; boolean that tell player win or lose that stage

Constructor

+ AfterStageScene(boolean Pass)

Getter

+ int getState();

Method

+ void paint(Graphics g); to paint this scene
+ void updateLogic(); update each parameters

Interface "game.gamescene.GameScene"

interface for every scene of this game

void updateLogic(); update each parameters

int getState(); return state

Class "game.gamescene.HelpPanel" extends JComponent implements GameScene

this class is for creating help panel in main menu

Field

- static Font header; font for header
- static Font other; font for other
- String leftkey; string to draw
- String rightkey; string to draw
- String normalbullet; string to draw

- String piecingbullet; string to draw
- String icybullet; string to draw
- String pausekey; string to draw

Constructor

+ HelpPanel()

Method

+ void paint(Graphics g); paint this class
+ void updateLogic();
+ int getState();

Class “game.gamescene.SettingPanel” extends Jcomponent implements GameScene

this class is for creating option panel in main menu, play also config key and mute sound in this panel

Field

- static Font header; font for header
- static Font other; font for other
- String leftkey; string to draw
- String rightkey; string to draw
- String normalbullet; string to draw
- String piecingbullet; string to draw
- String icybullet; string to draw
- String pausekey; string to draw
- boolean[] set; array of boolean that will enable which keyBinding will be set

Constructor

+ public SettingPanel();

Method

+ void paint(Graphics g); paint this class
+ void updateLogic(); update all parameters
+ getState();

Class “game.gamescene.StageScene” extends JPanel implements GameScene

this class is for generating stage scene

Field

+ static StageScene instance; global variable for StageScene for those entities can call this
+ Player player; player
Δ static final AlphaComposite transcluentBlack;
Δ static final AlphaComposite opaque;
- WaveGameState waveGameState[]; state of each wave, it will be the game scene of one wave,
- WaveController waveController[]; It’s runnable file to control each WaveGameState
- Thread thread[]; Collect thread for each wave
- int currentWave; current wave

Constructor

+ StageScene(int Stage, int subStage);

Getters / Setters

+ WaveGameState getCurrentWave();
+ synchronized Player getPlayer();
+ synchronized void setPlayer(Player player);
+ synchrnoized int getState();

Method

+ void updateLogic(); will check if game will be paused or not and will pause the game, also check if stage is completed or not

+ void paint(Graphics g); paint this class
+ void stateAllThread(); start all thread
+ void gameover(); if health is zero, will switchScene to AfterStageScene

Class "game.gamescene.TitleScene" extends JPanel implements GameScene

this class is for creating main menu, chapter select scene and shop

Field

+ static final int CHAPTER_SELECT; Integer to reminds state number
+ static final int SHOP; Integer to reminds state number
+ static final int MAINSCENE; Integer to reminds state number
+ static final int HELP; Integer to reminds state number
+ static final int OPTION; Integer to reminds state number
+ static final TitleScene instance; global variable for TitleScene to be called by any entities
- static AlphaComposite notOver; AlphaComposite when mouse is not over
- static AlphaComposite Over; AlphaComposite when mouse is over
- static Font goldFont; font for drawing remaining gold in shop
- static Font itemFont; font for drawing item's name in shop
+ int state; state to select which pictured will be shown
- int upgradeBarY[]; position of each Y axis for drawing upgradable gauge
- int itemX[]; position of each item string
- int itemY[]; position of each item string
- String itemName[]; string for each item name
- boolean shop[]; boolean for what player's going to buy
- SettingPanel OptionPanel; OptionPanel for main menu
- HelpPanel HelpPanel; HelpPanel for main menu
- BufferedImage s[][]; image of each state
- BufferedImage cs; current scene

- BufferedImage BG; background image
- BufferedImage Play1; a button for this scene
- BufferedImage Play2; a button for this scene
- BufferedImage Exit1; a button for this scene
- BufferedImage Exit2; a button for this scene
- BufferedImage Help1; a button for this scene
- BufferedImage Help2; a button for this scene
- BufferedImage Option1; a button for this scene
- BufferedImage Option2; a button for this scene
- BufferedImage Play; a button to be drawn choose between Play1 or Play2
- BufferedImage Exit; a button to be drawn choose between Exit1 or Exit2
- BufferedImage Help; a button to be drawn choose between Help1 or Help2
- BufferedImage Option; a button to be drawn choose between Option1 or Option2
- BufferedImage LV_MAX; a button that will be drawn when upgrade reached max level
- BufferedImage Levelled; will be drawn if a level is bought

Constructor

+ TitleScene()

Getters / Setters

+ static getInstance();

Method

+ void paint(Graphics g); paint this scene
+ void drawCost(Graphics2D g, int level, int y); drawing cost calculated from level
+ void buy(int g, int i); buy item

+ void updateLogic(); update current scene from state and check if mouse is over any button, it will glows, also play sound and enter other scene
+ int getState();

Class "game.gamescene.WaveController" implements Runnable

this class is for controlling all wave in each stage by Thread

Field

- WaveGameState wave; keep wave state will be updated
- Thread prevThread; previous thread that tell others to wait this thread before running

Method

+ synchronized vlkdtogglePause(); use notifyAll(); to pause or unpause game

```
public synchronized void togglePause() {
    if (prevThread == null || !prevThread.isAlive()) {
        wave.setPause(!wave.isPause());
    }

    notifyAll();
}
```

+ void run(); while a thread is still running, other will wait until it finished to run after that by using join();

```
@Override
public void run() {
    try {
        if (prevThread != null)
            prevThread.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    while (true) {
        try {
            Thread.sleep(GameManager.REFRESH_DELAY);
        } catch (Exception e) {
            e.printStackTrace();
        }
        synchronized (this) {
            if (wave.isPause()) {
                try {
                    wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
        GameManager.getGameScene().repaint();
        wave.updateLogic();
        wave.clearDestroyed();
        /*if (wave.clearDestroyed()) {
            break;
        }*/
        if (wave.getState() == WaveGameState.DEAD) {
            break;
        }
    }
}
```

Class “game.gamescene.WaveGameState” extends JComponent implements GameScene

this class is for generate all wave in stage and control whether scene should go in next state

Field

+ static final int SPAWNING_ENEMY; a constant number to remind state what state is this number is
+ static final int READY; a constant number to remind state what state is this number is
+ static final int FINISH; a constant number to remind state what state is this number is
+ static final int GAMEOVER; a constant number to remind state what state is this number is
+ static final int DEAD; a constant number to remind state what state is this number is
- String waveName; name of wave
- Background backGround; background
- PlayerStatus playerStatus; player status
- StageScene stage; stage
- Label flag; word at beginning of stage
- CopyOnWriteArrayList<EnemyGroup> enemyGroups; list of enemyGroups
- CopyOnWriteArrayList<Bullet> playerBullets; list of playerBullets
- CopyOnWriteArrayList<Bullet> enemyBullets; list pf enemyBullets
- CopyOnWriteArrayList<Particle> particles; lists of particle
- int WaveState; state of wave
- int delay; delay before wave ended
- boolean pause; paused or not paused

Constructor

+ WaveGameState(StageScene stage, String waveName,

CopyOnWriteArrayLust<EnemyGroup> enemyGroups);

Getters / Setters

+ boolean isPause();

+ void setPause(boolean flag);

+ String getWaveName();

Method

+ void updateLogic(); check collision and update logic of this class

+ void paint(Graphics g); paint this class

+ void addEntity(Entity entity); add entity to list

+ void addParticle(Particle particle); add particle to list

+ void clearDestroyed(); clear destroyed unit

+ int getState(); return WaveState

+ void forceEnd(); end this wave

+ boolean isGameOver(); return true when game over

+ String toString(); return string on label

Class ‘‘game.GameManager’’

this class is for managing all class for running this game

Field

+ static final int WIDTH; screen’s width

+ static final int HEIGHT; screen’s height

+ static final int REFRESH_DELAY; delay before refresh

+ static final int TICK_PER_SECONDS; tick per second

- private static GameWindow gameWindow; current window

Method

+ static void runGame(); run game by all class

+ static JComponent getGameScene(); get current scene

+ void switchScene(GameScene scene); switch game scene

Class "game.GameWindow" extends JFrame

this class is for creating JFrame for this game, and also add action listener

Field

- static GameWindow world; instance of GameWindow
- GameScene currentScene; current scene of that window

Constructor

Δ GameWindow();

Getters

+ getCurrentScene

Method

+ void switchScene(GameScene scene); switch scene
- void addListener(); add all listener to this JFrame and will be set on utility.InpuUtility

Class "exception.FileFormatException" extends Exception

this class is for exception that created for wrong file format opened

Constructor

+ FileFormatException(); draw error pane

Class "exception.FileMissingException" extends Exception

this class is for exception that created for missing file

Constructor

+ FileMissingException(String message); draw error pane

Class “entity.particle.BombParticle” extends Particle

this class is for creating bomb particle

Field

- int counter; for updating frame

Constructor

+ BombParticle(double x, double y);

Getter

+ getFrame()

Method

+ draw(Graphics2D graphics2d); draw this animation

+ boolean isVisible();

+ int getZ();

+ void update(); update this animation

+ BufferedImage getImage();

+ int getAct();

Class “entity.particle.fireParticle”

this class is fore creating fire particle

Field

- Player player; player

- int counter; for updating frame

- int direction; direction for this particle

Constructor

+ fireParticle(Player player, double x, double y)

Getter

+ int getFrame()

Method

+ boolean isVisible()

+ int getZ();

+ void update(); update animation

+ void draw(Graphics2D graphics2d); draw this animation

+ BufferedImage getImage(); return animation on that frame

+ int getAct();

Class “entiy.Particle” implements IRenderable

this class is for creating and generate all frame for every particle in this game

Field

Δ BufferedImage[] animation; array of of all frame for animation

Δ int[] frameDuration; duration for each frame

Δ int frame; frame number

Δ int state; this state

Δ double x; position of x

Δ double y; position of y

Constructor

+ Particle(double x, double y);

Method

+ boolean isDestroyed(); returned true if destroyed

Class “entity.enemy.Enemy”

this class is for managing all groups of enemies (e.g. move, slow, not slow and destroy)

Field

Δ double speed; enemy’s speed

Δ double nextX; x in next state

Δ double nextY; y in next stage

- double normalSpeed;

Constructor

+ Enemy(double health, double x, double y, double nextX, double nextY, double speed)

Getters / Setters

+ setSpeed(double d);
+ getSpeed();
+ setNextX(int nextX);
+ setNextY(int nextY);

Method

+ getZ();
+ abstract move();
+ slow(); slow enemy
+ deSlow(); recover enemy from being slowed
+ isInRightPos() check if enemy reach end point before continue moving

Class "entity.enemy.EnemyGroup"

this class is for control enemies in group (e.g. they will run as a looped or slowed together)

Field

+ static int MOVE; state that enemies moving,
+ static int HOLD; state, that enemies not moving
- ArrayList<Enemy> enemies; list of enemies
- ArrayList<Point> position; list of position
- int direction; direction of enemy
- int numEnemy;
- boolean looped;
- boolean ready;

- int state;
- int shootCounter;
- int shootDelay;
- int holdPositionCounter;
- int holdDuration;
- int slowCounter;

Constructor

+ EnemyGroup(ArrayList<Enemy> enemies, ArrayList<Point> position, int moveDuration, boolean looped);

Getters

+ isReady()
+ getEnemies()

Method

+ void update(); control enemy hold or move to next position
- void hold(); control holding for enemy
- void doShoot(); random enemies that will shoot
+ void shifGroup(int dx, int dy); move every point by dx and dy
+ void draw(Graphics2D g); draw this scene
+ clearDestroyed(); clear
+ slow(double slow, int slowTime), slow enemy

Class "entity.enemy.OneEnemy"

this class is for generating single enemy for shoot, move or destroyed

Field

- static final int shootDelay; delay for shooting
- int shootCounter; counter for shoot delay

Constructor

+ OneEnemy(double health, double x, double y, int nextX, int nextY);

Method

+ void update(); update this enemy
+ void hit(Bullet hitBullet); if it was attacked, decrease hp or destroy
+ void move(); move or hold
+ void attack(); attack when shootCounter = shootDelay
+ void destroy(); destroy this enemy and show destroying animation
+ void draw(Graphics2D graphics2d); draw this enemy
- void updateFrame(); update frameCounter

Class "entity.bullet.Bullet"

this class is for normal bullet

Field

- int moveCounter; delay before move
- int moveDuration; delay per move
- int speed; speed of this bullet
- double damage; damage of this bullet
- Entity attacker; entity that shoot this bullet

Constructor

+ Bullet(double x, double y, double damage, int speed, Entity attacker);

Getters

+ double getDamage();
+ Entity getAttacker();

Method

+ void update(); move bullet if not destroyed
+ void draw(Graphics2D graphics2d); draw this bullet
+ int getZ();
+ void destroy(); destroy this bullet
- void move(); decrease moveCount or move

- void updateFrame();
- void destroyByOutofScreen(); to check if out of screen, bullet will be destroyed

Class “entity.bullet.IceBullet” extends Bullet

this class is for icy bullet of player

Field

- double slow; slow ratio

Constructor

+ IceBullet(double x, double y, double damage, double slow, int speed, Entity attacker);

Method

+ draw(Graphics2D graphics2d); draw this bullet
+ destroy(EnemyGroup hitted); if enemy got shotted, its group will be slowed

Class “entity.bullet.PiecingBullet” extends bullet

this class is for piecing bullet of player

Constructor

+ PiecingBullet(double x, double y, double damage, int speed, Entity attacker);

Class “entity.BackGround” implements IRenderable

this class is for painting background for each stage

Field

- BufferedImage background; background

Constructor

+ Background(StageScene stage);

Getter

+ getImage();

Method

+ draw(Graphics2D graphics2d)
+ isVisible();
+ getZ();
+ update();
+ isDestroyed();
+ getAct();
+ getFrame();

Abstract Class “entity.Entity” implements IRenderable, ICollidable

this class is for controlling all entity to move, spawn, attack, got attacked and also destroyed

Field

+ static final int SPAWNING;
+ static final int READY;
+ static final int ATK;
+ static final int HIT;
+ static final int HIT_ATK;
+ static final int DESTROYING;
+ static final int DESTROYED;
- BufferedImage[][] animation; images for animation
- int[][] frameDuration; array of frame duration
- Polygon[][] collisionBox; box that will be the area that enemies will be shotted
- int frameCounter; for update frame every frameDuration
- int act; action of entity

- Δ int frame; frame of that act
- Δ double health; entity's health
- Δ double x; position in x-axis
- Δ double y; position in y-axis
- Δ boolean willAttack; boolean that will tell whether it will attack or not
- Δ int state; state of this class
- \- boolean endAnimation; true when animation finished

Constructor

- \+ Entity(double health, double x, double y)

Getters / Setter

- \+ double getHealth();
- \+ BufferedImage getImage();
- \+ void setAct(int act)
- \+ int getAct()
- \+ int getFrame()

Method

- \+ boolean collideWith(ICollidable obj); Check if two things collide
- \+ abstract destroy(); destroy that entity and play animation
- \+ isDestroyed(); check if it's destroyed or not
- \+ isReady(); check if it's ready to move or not
- \+ isVisible();
- \+ setReady(); set Ready
- \+ setAttack(); set Attack
- Δ setAnimation(BufferedImage[][] animation); set Animation
- Δ setFrame(int frame); setFrame
- Δ isEndAnimation(); check is isEndAnimation or not
- Δ getNumberFrame(); return number of frame
- Δ willAttack(); set willAttack to true

Interface “entity.ICollidable”

interface for object that collide with other (player’s bullet and enemies or enemies’ bullet and player)

int getX(); get x-axis position

int getY(); get y x-axis position

boolean collideWith(ICollidable obj); to check if this collide with object obj or not

void setCollisionBox(Polygon[][] collisionBox); to set the area that will be used to check if object is collide with others or not

Polygon getCollisionBox(); return area to check if this object is collide with others or not

Interface “entity.IRenderable”

this interface is for object that can be rendered or drawn

void draw(Graphics2D graphics2d); to draw this

boolean isVisible(); to check this is visible or not

int getZ(); to get the order of what should be draw first

void update(); to update parameters of this

boolean isDestroyed(); to check if this is destroyed or not

BufferedImage getImage(); to get this image

int getAct(); to get this act

int getFrame(); to get this frame

Interface “entity.IShootable”

this interface is for object that can shoot

void attack(); to attack

Class “entity.Label” implements IRenterable

this class is for generating and draw label (e.g. wave 1)

Field

+ static Font header; font for header
- String displayStr; string that displaying
- int act; action of Label
- int frame; this all frame
- int frameCounter; count till the frame changed
- int[][] frameDuration; duration before change
- boolean play; true when not pause

Constructor

+ Label(String displayStr);

Getters

+ int getAct();
+ int getFrame();
+ String getDisplayString();

Method

+ update(); update the label
+ draw(Graphics2D graphics2d); draw the label
+ isVisible();
+ getZ();
+ isDestroyed();
+ getImage();
+ nextAnimation(); set play to true when act is 0

Class “entity.Player”

this class is for player, calculating whether player can move, attack or destroyed

Field

- int shootCounter; number that will tell when player will be able to shoot the bullet again
- int fireRate; number that will tell how frequently player can shoot the bullet
- int moveCounter; delay before player can move again
- int moveDuration; duration that took player to move
- double speed; movement speed of this player
- double damage; damage of normal bullet of this player
- double damagePiecing; damage of piecing bullet of this player
- double damageIce; damage of icy bullet of this player
- double slowIce; slow rate of icy bullet of this player
- ArrayList<Particle> particles; list particles that will be drawn

Constructor

+ Player();
+ Player(int health, double speed, double damage, double damagePiecing, double damageIce, double slowIce, int fireRate);

Method

+ update(); to run and check whether player can attack, destroyed or need to be spawn again
+ attack(); to fire one type bullet if available (Check by isShootReady()) or reduce shootCounter
- updateFrame(); to update frame of this player if he or she is destroyed or move
- move(); to move if able to (checked by moveDuration)

+ hit(); to call method destroy() this player when got hit
+ destroy(); to destroy spacecraft
+ draw(Graphics2D graphics2d); to draw this class
+ getZ();
+ isShootReady(); to check is this player able to shoot by shootCounter

Class "entity.PlayerStatus"

this class if for drawing player status in the bottom of screen

Field

- static AlphaComposite halpha; composite for health bar
- static AlphaComposite notOver; composite for pause button
- static Font font; font for player status
- StageScene stage; scene of this stage
- boolean isOver; boolean for pause button
- int x; position of mouse in x-axis
- int y; position of mouse in y-axis

Constructor

+ PlayerStatus(StagScene stage);

Method

+ draw(Graphics2D graphics2d); to draw this player status
+ update(); to check if mouse is over pause button or not
+ isVisible();
+ getZ();

Class Main

this is the main class of this game

Method

+ static void main(String[] args);