**Project: Type Wizard**

This game is a Role-Playing-Game. Player is a wizard that can use spell by typing spell in different word. This game is Endless Mode that player can play until the wizard die, wizard has level and status that affected to his strength ,for the enemies it’s strength increasing by the level of the wizard too.

Control

Left Arrow : Move Left

Right Arrow : Move Right

Up Arrow : Jump

Esc : Pause/UnPause

While Pause > Q : to Main Menu

X : mute/unmute music

H : show Leader Board

Skill > F : fire skill , single-target ,burn debuff (dmg/sec)

I : ice skill , single-target , slow debuff (speed-)

M : meteor skill , multi-target , huge damage

P : poison skill , multi-target , poison debuff (dmg/sec)

S : spike skill , multi-target , push skill low damge



Figure 1 showing main menu. You can press any key to start the game.



Figure 2 showing map and user interface in game.



Figure 3 showing how wizard cast spell.



Figure 4 showing monster Yeti get hit by meteor skill.



Figure 5 showing monster Yeti effected by Freezing de-buff.



Figure 6 showing monster Yeti has fallen.

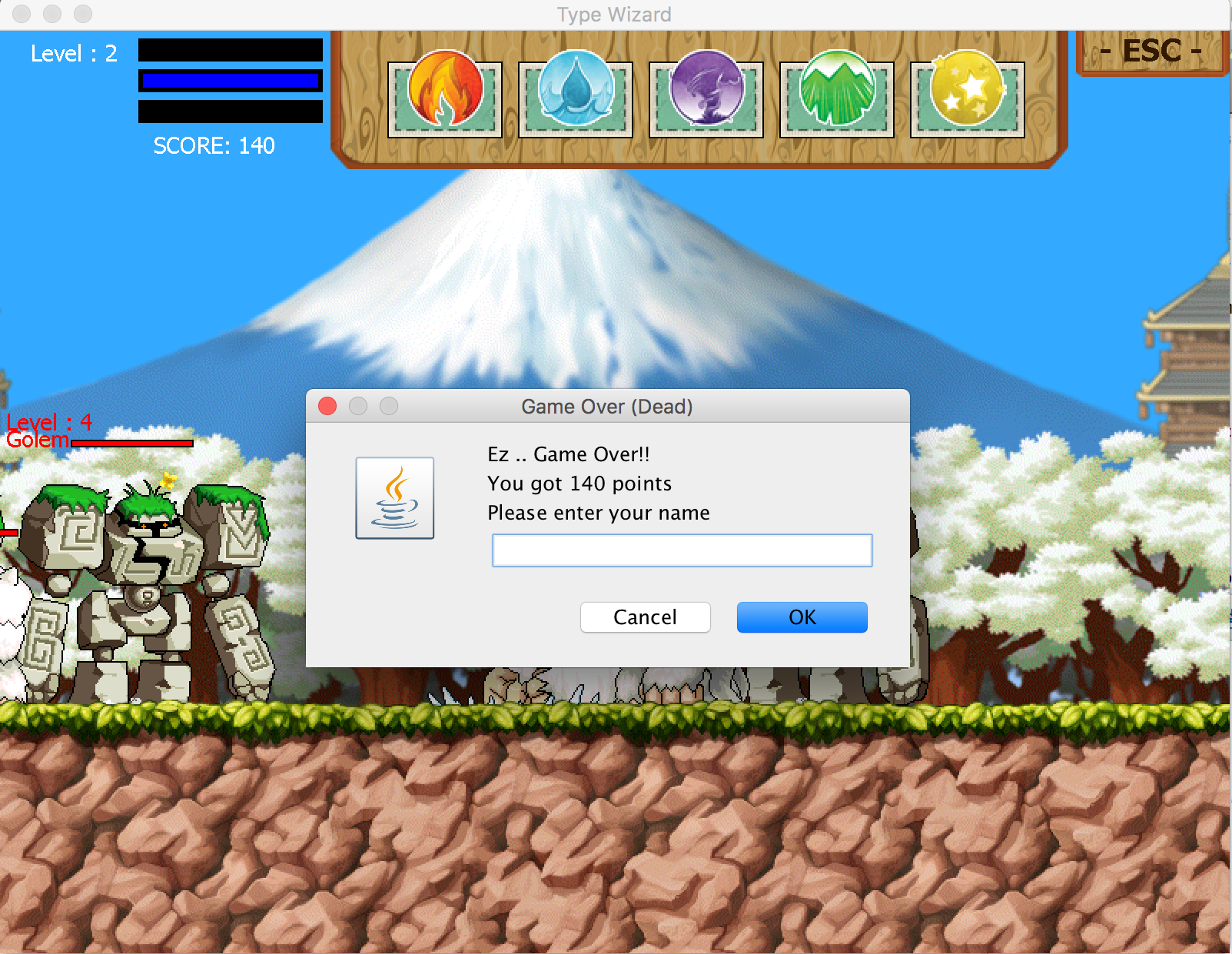


Figure 7 showing GameOver Screen.

**Package input**

**Class InputUtility**

Handle all of keyboard events occur during the game is running.

Field

* boolean[] keyPressed; 256 key that collects user typing
* String spell; empty string declare
* boolean escTriggered; to collect ESC key Triggering initial is false
* boolean escPressed; to collect ESC key Pressing initial is false

Constructor

* Default Constructor

Method

* boolean getKeyPressed(int key) ; getter for keyPressed Array
* void setKeyPressed (int key, boolean keyPressed) setter for keyPressed Array
* void clearSpell() reset the String spell to the beginning
* void reset () reset keyPressed Array to the beginning

**Package render**

**Class GameScreen**

User contact ,check all of renderable object and draw it if it visible.

Field – (none)

Constructor

* GameScreen(); Initialize size and add KeyListener

Method

* void paintComponent(Graphics g); Paint all components in this Class

**Class GameTitle**

Field

* boolean isTitle ; to identify what screen should be show
* boolean swap ; use in swapping screen process
* int tick ; general time count in class
* int R = 255, G = 255, B = 255; color of bottom label
* JLabel start; label that show in bottom

Constructor

* GameTitle(); Initialize Class in a proper way

Method

* void paintComponent(Graphics g); paint all components in class
* void update(); update all values in class
* Getter & Setter

**Interface Renderable**

Initialize important method that class need to have if it can Render.

* void draw(Graphics g) ;
* boolean isVisible();
* Boolean getZ();

**Class RenderableHolder**

Field

* RenderableHodler instance ; instance of Class
* List<Renderable> entities ; store many object that can render

Method

* static RenderableHolder getInstance(); get this class
* void add(Renderable o); add object to entities
* void remove(Renderable o); remove object from entities
* void removeAll();remove All objects from entities
* List<Renderable> getRenderableList(); getter for entities

**Class Resource**

All of Resource that use in game include font,picture and sound files.

Field

* int screenWidth, screenHeight ;
* Font wordFont,standardFont,biggerFont,pauseFont,skillFont;
* BufferedImage hero, hero\_f, hero\_a, hero\_af, hero\_d, hero\_df ;
* BufferedImage monster\_golem\_1\_1, monster\_golem\_1\_2, monster\_golem\_1\_3, monster\_golem\_1\_1\_f, monster\_golem\_1\_2\_f, monster\_golem\_1\_3\_f;
* BufferedImage monster\_yeti\_1\_1, monster\_yeti\_1\_2, monster\_yeti\_1\_3, monster\_yeti\_1\_4, monster\_yeti\_1\_1\_f, monster\_yeti\_1\_2\_f, monster\_yeti\_1\_3\_f, monster\_yeti\_1\_4\_f;
* BufferedImage statusBG, skillBoard, pauseBox, banner;
* BufferedImage land1, background1, titleBG;
* BufferedImage ice1, fire1, meteor1, meteor1\_2, poison1, spike1;
* BufferedImage skillBoxBG, fireStatus, iceStatus, meteorStatus, poisonStatus, spikeStatus;
* BufferedImage burn, poison, freeze;
* AudioClip titlebgm, screenbgm, hit, jump, levelup, score, iceskill, meteorskill, meteorskill2, fireskill, poisonskill, spikeskill

Method

* BufferedImage getImage(String directory)
* AudioClip getSound(String directory)

**Package entity**

**Abstract Class Moving**

Field

* int x,y ; the most important thing to identify drawing coordinate

Constructor

* Moving(int x, int y); Initailize x,y that important to identify drawing coordinate

Method

* abstract void update(); update all value in a correct way

**Class Land**

Field

- int xc; use to compare distance and predicted action

- boolean isStart,isEnd; to identify where hero can’t go on anymore

Constructor

* Land(int x, int y); Initailize x,y that important to identify drawing coordinate

Method

- void update(); update all value in a correct way

- Getter & Setter

- void draw(Graphics2D g); draw All Components in class

- boolean isVisible();to identify that it would show in screen or not

- int getZ(); to identify priority of drawing

**Class Background**

Field

- int xc; use to compare distance and predicted action

- boolean isStart,isEnd; to identify where hero can’t go on anymore

Constructor

* Background(int x, int y) Initailize x,y that important to identify drawing coordinate

Method

- void update() update all value in a correct way

- Getter & Setter

- void draw(Graphics2D g); draw All Components in class

- boolean isVisible(); to identify that it would show in screen or not

- int getZ(); to identify priority of drawing

**Class Hero**

Field

* int frameCount, count, countA, frameCountA, direction , temp, i , countD , frameCountD, tick, gravity, velocity, STR, INT, level, hp, attack, mana, manaTick , maxMp, maxHp, notBeHitCount’
* boolean isJumped, isDead, isOutOfMana, isHitting, isRight, isLeft, isStop, isSkill, isCasting;
* Land land; Land Controller object
* Background background; Background controller object
* HeroStatus heroStatus; Object that draw HP,MP,XP bar
* Skill[] skills ; skills that hero can use it has 5 different skills;
* Word[] words; words that hero need to type for casting skills
* Monster nearMon; Nearest monster from hero

Constructor

* Hero(int x,int y, Land land, Background background, HeroStatus heroStatus); Initialize Hero in a correct way

Method

* void update(); Call all of these below method & others in a particular way
* void checkLevel(); to check that is hero has XP to involve
* void setHpMp(); setter for both HP&MP
* void deadAnimation(); call when hero die to shows dead animation
* void idleSpellAnimation();method to make spell casting animation
* void idleAnimation();method to make hero idle animation
* void jump(); method to make hero move
* void walkLeft(); method to make hero move
* void walkRight(); method to make hero move
* void castingSkill(); method to make skill animation
* void checkCastingSkill(); to check that hero is now free or not
* void removeMonster(); remove dead Monster from map
* Getter & Setter

**Class GameLogic**

Field

* Land land; initialize land controller object
* Hero hero; our wizard has been born here. (initialize hero object)
* HeroStatus heroStatus; Top Right Status show level,HP,MP and XP
* SkillStatus skillStatus; Top Banner that shows skill in used
* boolean castFail , iCast, fCast, mCast, pCast, sCast, isPause ; Status what wizard is casting now
* Word ice, fire, meteor, poison, spike; initialize word to compare input
* int tick = 0, speed; collect tick and speed of thread sleep
* Background background; Background Controller Declare
* Banner pauseBanner; PauseBanner Object Declare
* GameTitle title; GameTitle Object Declare
* Thread t1,t2; Dual Thread,Handle SoundControl & Pause
* BgmControl bgm; Background Sound Controller

Constructor

* GameLogic(GameTitle gameTitle) Initial Game Logic

Method

* void update(); update land, background, hero, heroStatus, skillStatus add to RenderableHolder
* Getter & Setter

**Class Word**

Field

* String[] words; to collect each word
* int[] x; collect each letter in spell word
* boolean[] press; collect which skill player will use
* boolean isVisible; collect visible for checking to draw

Constructor

* Word(String s) Initial Word Object

Method

* void draw(Graphics2D g); draw word spell when player pressed keyboard(if not equal it will disapper)
* boolean isEqual(Word w); check length of this word and w if equal return true
* boolean cast(String s); if the string s is equal to spell word return true
* boolean isVisible(); check to draw
* int getZ(); Return Z
* Getter & Setter

**Class BgmControl**

this Class is Runnable that control sound bgm when playing a game

Field

* boolean isPause; check to stop or not

Method

* void run(); run by thread to play/stop sound bgm when playing a game

Getter & Setter

**Package entity.skill**

**Abstract Class Skill**

Field

* boolean isPlaying; collect Playing to check when to play the – animation
* int x, y; skill location
* int frameCount, count; Counter of Animation
* int frameWidth, frameHeight; Animation size that will show from frameWidth \* frameHeight
* int attackRange; to collect the attack range to interact with monster
* int direction; 1 = Right, 2 = Left

Method

* abstract void update(); update animation of each skill
* void play(); set Playing to true and set frameCount,Count to 0
* void stop(); set Playing to false and set frameCount,Count to 0
* int getZ(); Return Z
* void setX(); set this.x -= x and set attack range = x in case that background move( right/left key is pressed and hero location not change)
* Getter & Setter

**Class FireSkill**

Constructor

* FireSkill(int x, int y,int direction); Initial Fire skill

Method

* void update(); update Fire skill animation
* void draw(); draw Fire skill
* boolean isVisible(); if Playing return true

**Class IceSkill**

Constructor

* IceSkill(int x, int y,int direction); Initial Ice skill

Method

* void update(); update Ice skill animation
* void draw();draw Ice skill
* boolean isVisible(); if Playing return true

**Class MeteorSkill**

Field

* int frameCountX, frameCountY; Counter of Meteor Animation

Constructor

* MeteorSkill(int x, int y,int direction); Initial Meteor Skill

Method

* void update(); update Meteor skill animation
* void play(); set Playing to true and set frameCountX,Y to 0
* void stop(); set Playing to false and set frameCountX,Y to 0
* void draw(); draw Meteor Skill
* boolean isVisible(); if Playing return true

**Class PoisonSkill**

Constructor

* PoisonSkill(int x, int y,int direction); Initial Poison skill

Method

* void update(); update Poison skill animation
* void draw(); draw Poison skill
* boolean isVisible(); if Playing return true

**Class SpikeSkill**

Constructor

* SpikeSkill(int x, int y,int direction); Initial Spike skill

Method

* void update(); update Spike skill animation
* void draw(); draw Spike skill
* boolean isVisible(); if Playing return true

**Package entity.monster**

**Class Cage**

this class collect the monster in List

Field

* Cage instance;
* List<Monster> cage; to collect monster in List

Method

* Cage getInstance(); Return instance
* void add(String n, Land l, Hero h); add to cage
* void remove(int i); remove from cage
* void updateAll(); update all monster that in cage
* void removeAll(); remove all monster
* List<Monster> getCage();

**Abstract Class Monster**

Field

* int x, y; location of Monster
* int hp, maxHp; current HP and Max Hp
* int type; Monster’s Type 1=Immune, 2 = Ice
* int attack; Monster’s attack damage
* int level; Monster’s Level
* int direction; 1 = Left, 2 = Right
* int countWalk ,frameCountWalk, countDead, frameCountDead, countPanic; Counter of animation
* int damageTaken; damage that use by each skill
* double xHp; use to draw Monster’s Hp bar
* boolean isDead, isPanic; check Monster state
* Land land; collect land data
* Hero hero; collect hero data

Method

* abstract void hit(Hero hero, Skill skill); take damage from hero by each skill
* abstract void update();update currentHp and check Monster’s state
* void setX(int x); set this.x -= x in case that background move( right/left key is pressed and hero location not change)
* Getter & Setter

**Class Golem**

This monster is Immune type. It resists from debuff. Attack by walk

Constructor

* Golem(int x, int y, Land land, Hero hero)

Method

* void update(); update currentHp and check Monster’s state
* void deadAnimation(); when hp=0 animation dead active
* void panicAnimation(); when monster got hit. It will panic and nock back
* void walkAnimation(); Collect Tick of walk animation and transform monster location
* void walkRight(); When hero walks to the middle of the Screen. The Screen push front, monster will push back.
* void walkLeft(); When hero walks back the Screen push back ,monster will push front
* void draw(Graphics2D g); draw Monster Golem
* void hit(Hero hero, Skill skill); take damage from hero by each skill
* boolean isVisible(); check to draw
* int getZ(); Return Z

**Class Yeti**

this Monster is Ice type. It weak to fire. Use fire to take more Damage. Attack by Punch.

Field

* int countAttack, frameCountAttack, attackDirection, speedDecrease; Counter and effect from Freeze’s Debuff
* boolean isAttack; check if Yeti is attacking
* int[] debuff; Collect Debuff(have 3 debuff. Burn Freeze(slow) Poison)
* int[] countDebuff; Collect Tick of Debuff
* int[] timeCountDebuff ; Collect times of Debuff

Constructor

* Yeti(int x, int y, Land land, Hero hero); Initial Monster

Method

* void update(); update currentHp and check Monster’s state
* void deadAnimation(); When hp=0 animation dead active
* void panicAnimation(); When monster got hit. It will panic and nock back
* void attackAnimation(); When Yeti near the hero, it will change to attack state
* void walkAnimation(); Collect Tick of walk animation and transform monster location
* void walkRight(); When hero walks to the middle of the Screen. The Screen push front, monster will push back.
* void walkLeft(); When hero walks back the Screen push back ,monster will push front
* void debuffDamage(); Collect tick. Monster will take Damage every 40 tick for 3 times.
* void damageFromDebuff(Hero hero, int damage); Damage that taken from debuff.
* void draw(Graphics2D g); draw Monster Yeti
* void hit(Hero hero, Skill skill); take damage from hero by each skill
* boolean isVisible(); check to draw
* int getZ(); Return z

**Package ui**

**Class Banner**

* this class is runnable that control the game pause and resume

Field

* boolean isVisible; check visible

Method

* void draw(Graphics2D g); draw Banner
* boolean isVisible(); check to draw
* boolean setVisible(); set visible
* int getZ(); Return Z
* void run(); run by thread to pause/resume game

**Class HeroStatus**

* this class keep the hp,mp and xp for drawing.

Field

* int score, health, mana, level,maxHp,maxMp,maxXp,xp;
* double xHp,xMp,xXp;
* boolean isDead; if true means dead

Constructor

* HeroStatus(); initial HeroStatus

Method

* void update(); update Level if(Xp >= maxXp)
* void addXp(int x); gain Xp
* void setMaxXp(); MaxXp = level\*100
* void setMaxHp(int x); MaxHp = x
* void setMaxMp(int x); MaxMp = x
* void setCurrentHp(int x); hp = x
* void setCurrentMp(int x); mp = x
* void getScore; Return score
* void addScore(int score); Gain score
* void resetLevel(); Reset Level to level 1
* void getLevel(); Return level
* void changeLevel(); increase Level by 1.
* void draw(Graphics2D g); draw Hp,Mp,XP bar + draw Hero’s level
* boolean isVisible(); check to draw
* int getZ(); return z

**Class SkillStatus**

this class keep the cast state to draw which skill is going to activate.

Field

* double fAngle , iAngle, mAngle, pAngle, sAngle; angle of rotation of each skill
* boolean fCasting, iCasting, mCasting, pCasting, sCasting; Boolean that check which skill is Casting
* boolean isPause; check if Game is pause

Method

* void draw(); draw each skill status if casting add some rotation
* boolean isVisible(); check to draw
* int getZ(); return Z
* Getter & Setter