CP Election Documentation

Created by

Niti Assavaplakorn 6031031221
Natchapol Srisang 6031308121

2110215 Programming Methodology
Semester 1 Academic Year 2018
Chulalongkorn University

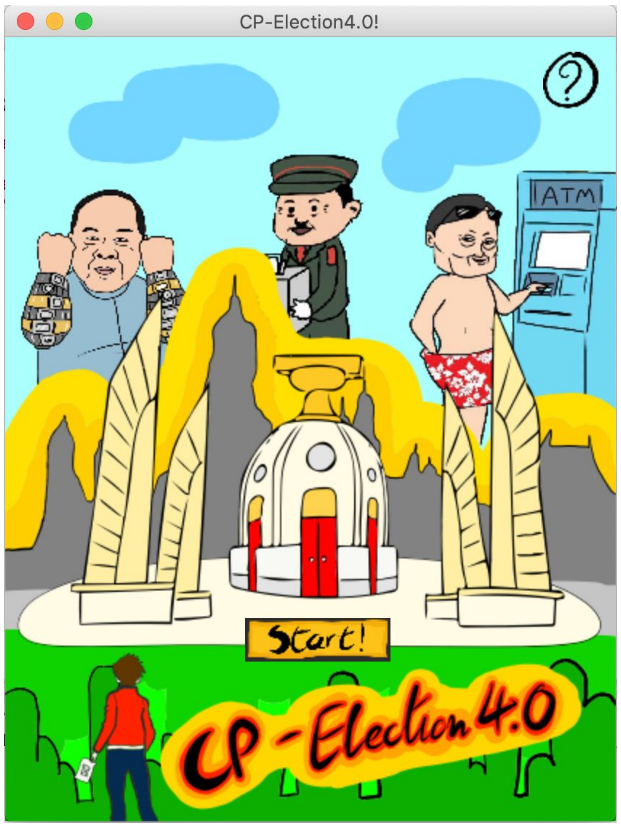
CP Election Documentation

Introduction

CP Election is a single player survival game. The goal is to fight against NPCs and finally beat the boss of the game. In addition to normal attack, the player can use various types of helpful(?) items dropped from NPCs to make the game easier.

User Manual

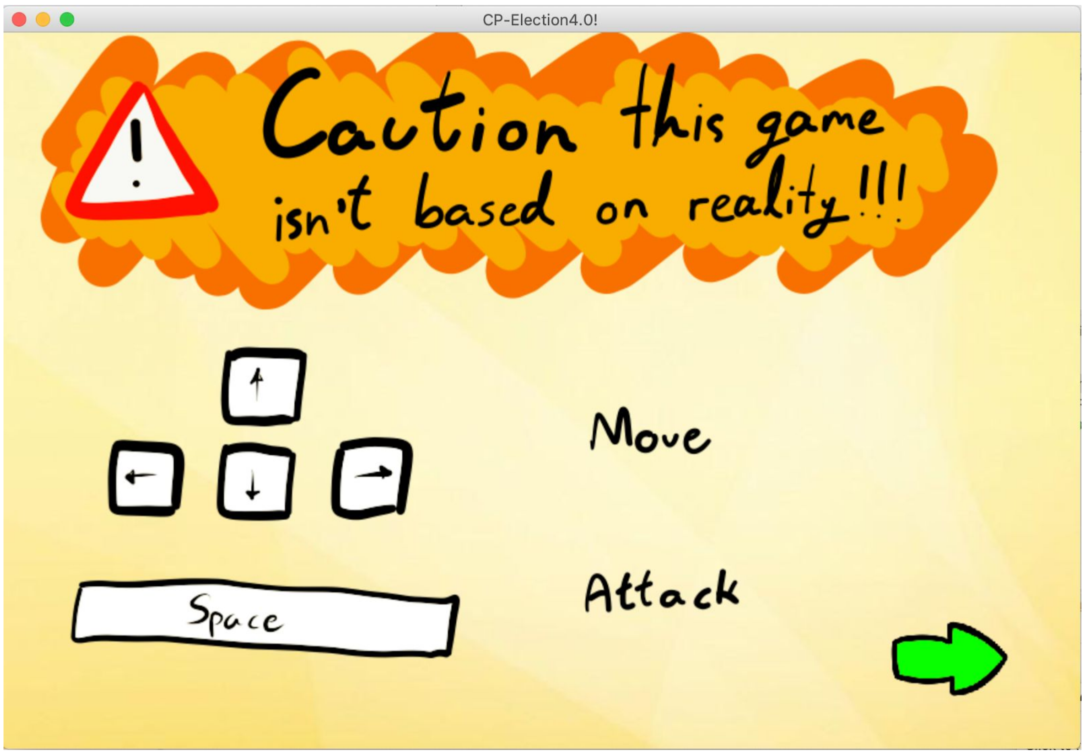
Start Scene

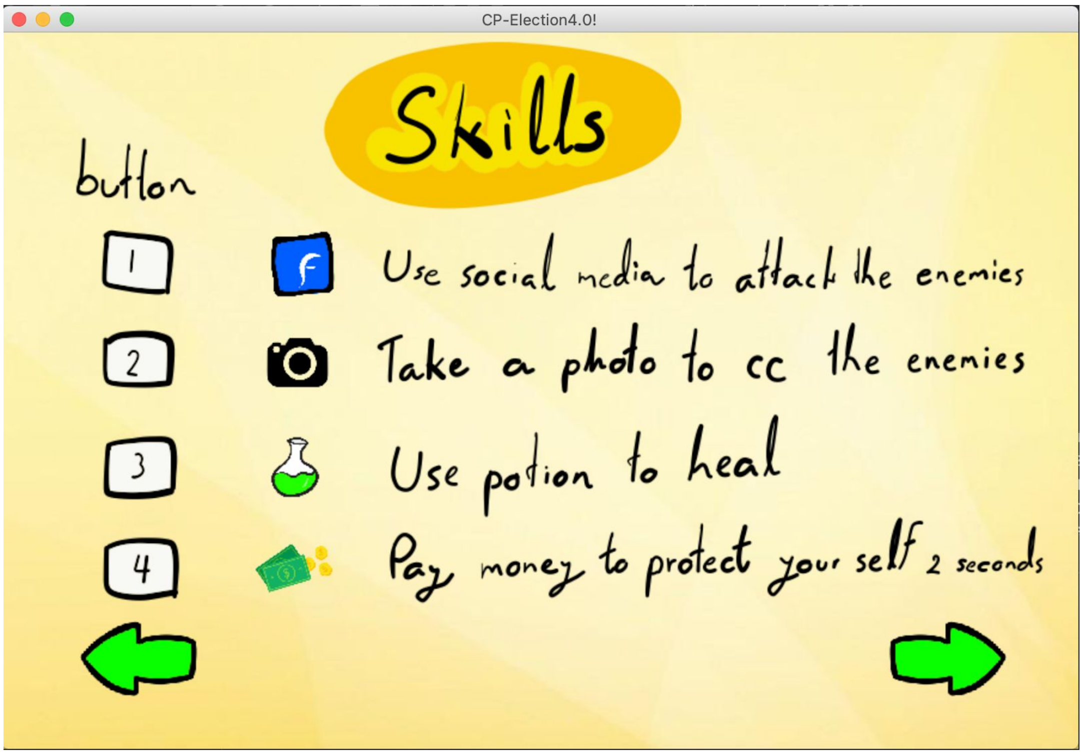


To start the game, click on the play button. For an instruction, click on the question mark button at the corner.

Instruction Scene

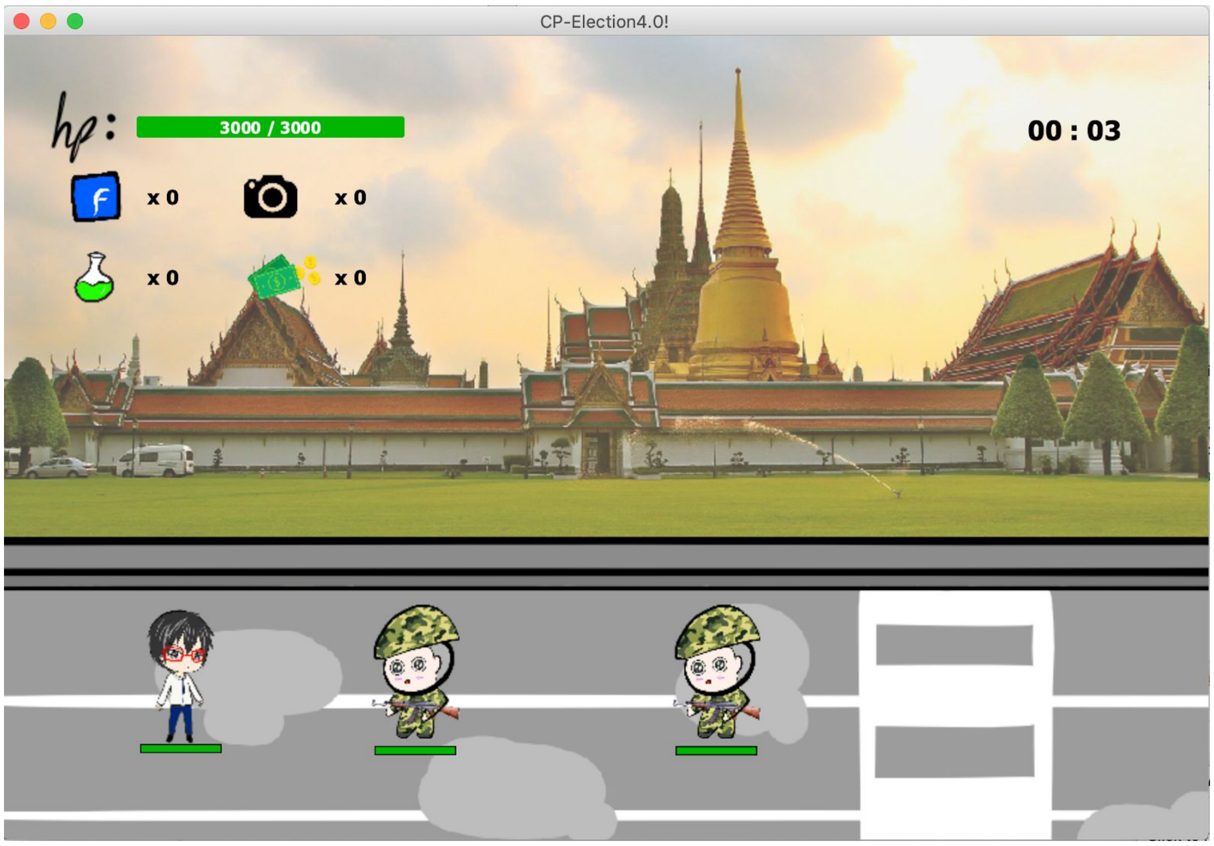
There are 2 pages in this scene both explain all control buttons and how does each item work.




CP-Election4.0!
Skills
button
1
Use social media to attack the enemies
2
Take a photo to cc the enemies
3
Use potion to heal
4
Pay money to protect your self 2 seconds

Game Scene

This game scene consists of 3 major components.



1. Main view

Main view is the view that shows everything in the current map. Background image, monsters, players and particles are shown in this area. Main view's area occupies the whole screen (including status bar area)

2. Status bar

Status bar is the place where the player's properties are shown. The following list are the properties that status bar shows.

- Player HP bar
- Inventory

3. Timer

The score of this game depends on the time can survive. So, there is a time indicator that shows total time that you have played the current session. Timer is shown on the top-right of the screen.

Game Control

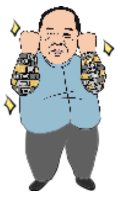
1. Movement

Move left: Arrow left
Move right: Arrow right
Move up: Arrow up
Move down: Arrow down
Attack: Space bar

2. Other controls

Use item: Number 1-4

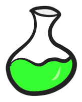
NPC

Image	Name	HP	Attack Damage	Spawn time	Skill
	Soldier	100	50 - 150	every 1.4 seconds	-
	Prawit	5,000	50 - 200	spawn after game started 30 seconds	Apply CC to player every 10 seconds

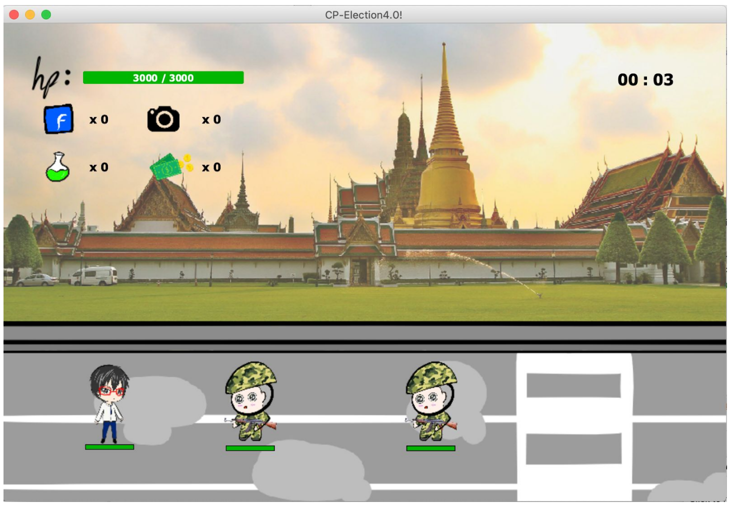
	Prayut	10,000	100 - 500	spawn after game started 1 minutes	Throw podium every 4 seconds (Podium can damage and stun player)

*** Prayut and Prawit aren’t affected by item.**

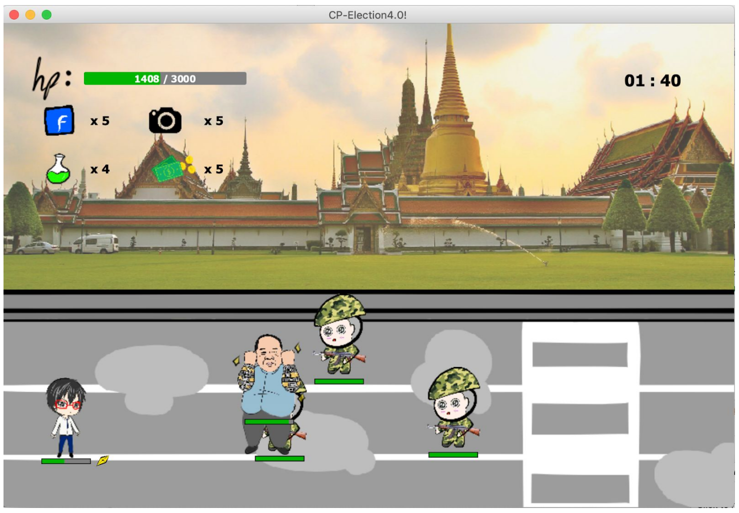
Items

Image	**Name**	**Use button**	**Maximum amount**	**Behavior**
	Attack	1	5	Damage 75 - 150 and knock back all npc on the map
	CC	2	5	Random CC type to all npc on the map.
	Heal	3	5	Heal player 200 - 500 HP
	Immune	4	5	Protect player from getting damage for 3 seconds.
	Revive	-	1	Revive player if player is dead.

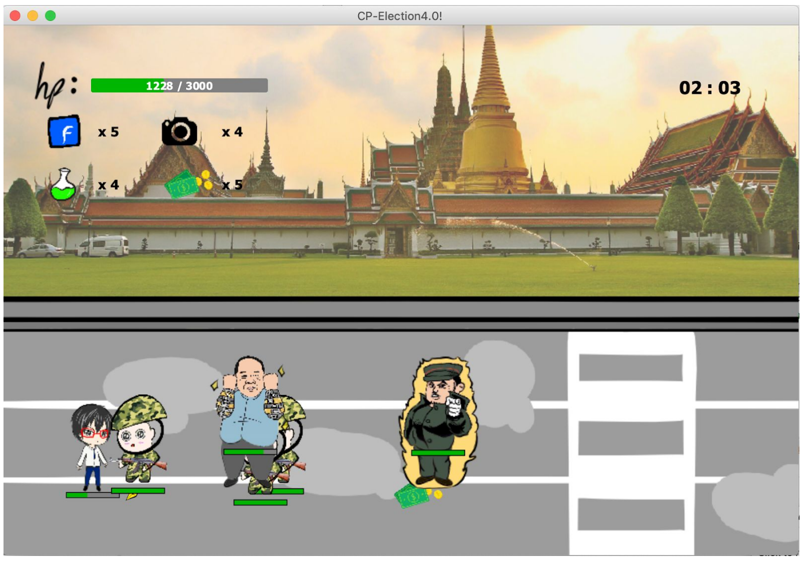
Gameplay



As soon as you have clicked "Start Game" button on the start scene, you will be presented in the map. You can control a player by using the guide above. As time passing, you will see NPC spawning. NPC will follow and attack you. Be cautious! When it touches you, you hurt. A simple way to attack NPC is to press space bar button on the keyboard. If you attack it repeatedly until it dies, it will drop item sometime.



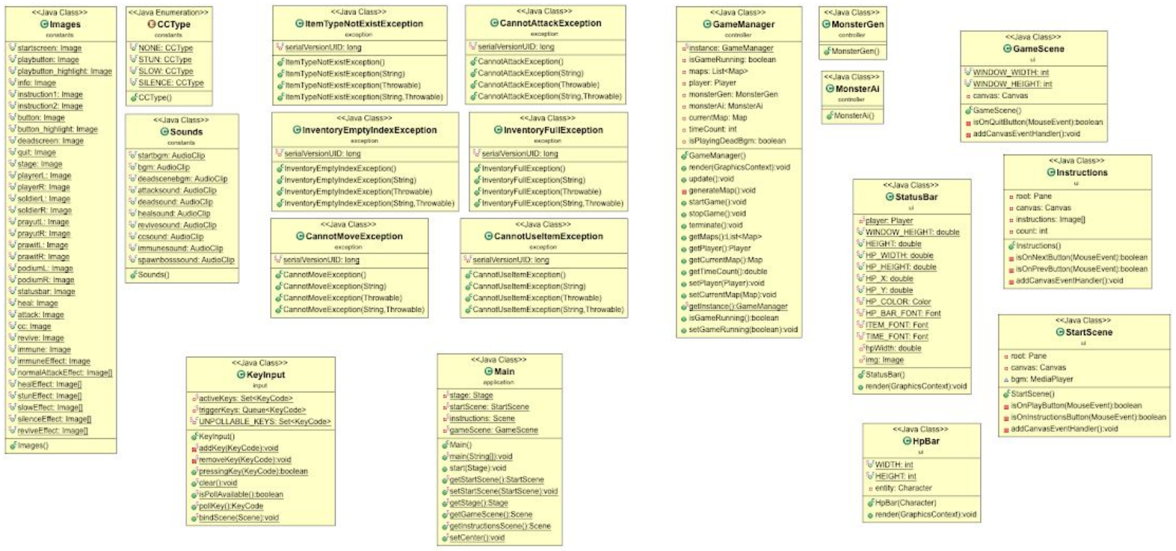
After game started 30 seconds boss prawit will spawn. He will start attack and CC you with his luxury watch that prize over 3 millions baht.

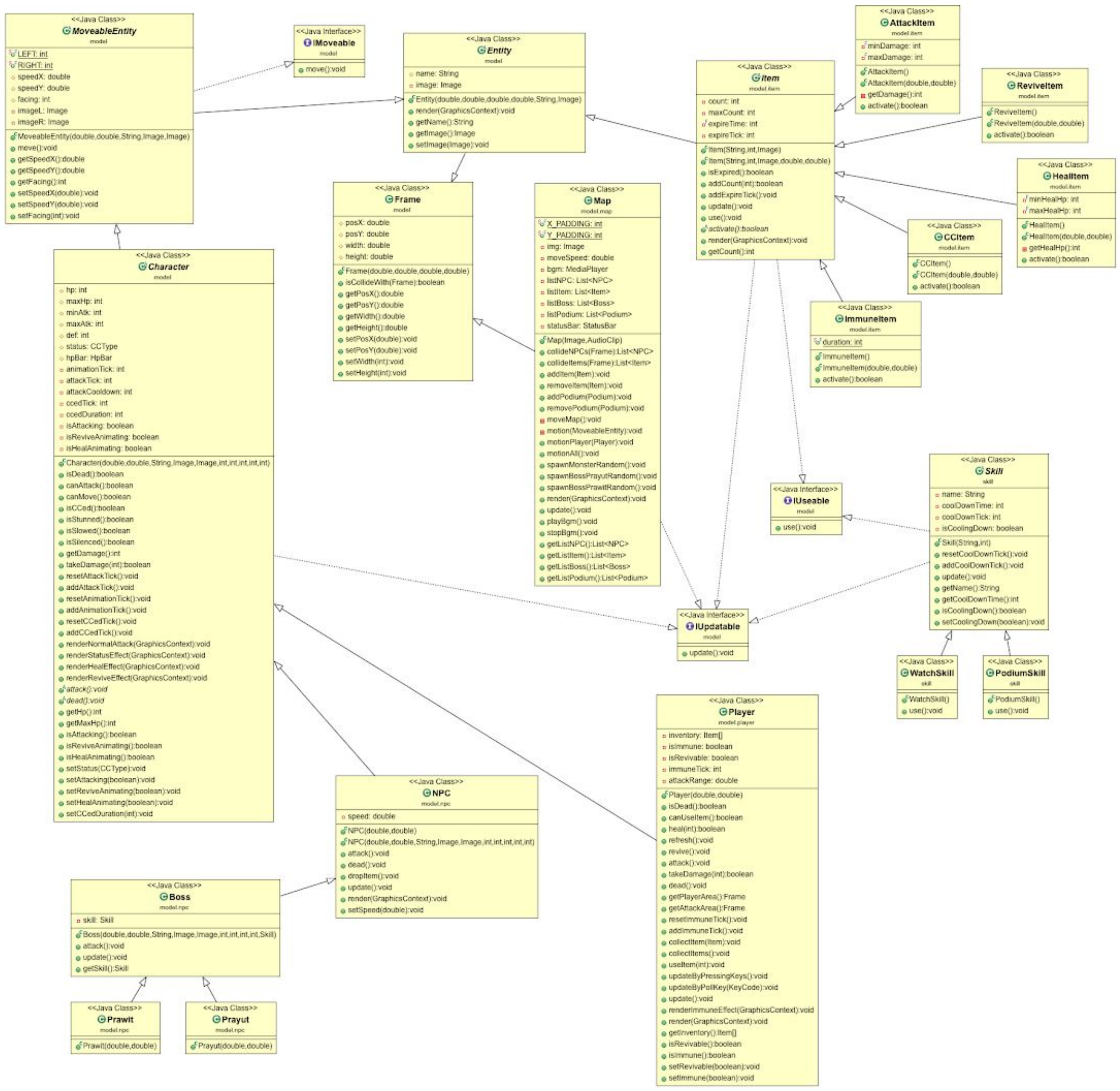


After game started 60 seconds boss Prayut will spawn and start to attack you. Be careful, he might throw podium at you if he is angry. Try to avoid the podium. It can damage and stun you too. Survive as long as possible!

!!! Good Luck !!!

Implementation Details





*** Noted that Access Modifier can be listed below**

+ (public)
# (protected)
- (private)
Underlined (static)
Italic (abstract)

1. Package application

1.1. Class Main

1.1.1. Fields

- Stage stage	Store the start stage.
- StartScene startScene	Store the start scene.
- InstructionsScene instructionsScene	Store the instruction scene.
- GameScene gameScene	Store the game scene.

1.1.2. Methods

+ void main(String[] args)	Launch the game.
+ void start(Stage primaryStage)	The main entry point of JavaFX application.
+ void setCenter()	Set the stage at the center of screen.
Getters of every fields & setter of startScene	

2. Package constants

2.1. Enum CCType (NONE, STUN, SLOW, SILENCE)

This enum contains all crowd control (CC) status types.

2.2. Class Images

This class contains all images.

2.2.1. Fields

+ Image startscreen	Store the image start screen background
+ Image playbutton	Store the image playbutton on start screen, set width to 100 and height to 30
+ Image playbutton_hilight	Store the image playbutton with highlight on start screen, set width to 100 and height to 30
+ Image info	Store the image info button on start screen
+ Image instruction1	Store the image of instruction page 1
+ Image instruction2	Store the image of instruction page 2
+ Image button	Store the image backbutton on instruction screen, set width to 100 and height to 66
+ Image button_highlight	Store the image backbutton with highlight on instruction screen, set width to 100 and height to 66
+ Image deadscreen	Store the image dead screen background
+ Image quit	Store the image quit button on dead screen
+ Image stage	Store the image background map, set width to 1000 and height to 667
+ Image playerL	Store the player left side image, set width to 66 and set height to 100

+ Image playerR	Store the player right side image, set width to 66 and set height to 100
+ Image soldierL	Store the npc left side image, set width to 66 and set height to 100
+ Image soldierR	Store the npc right side image, set width to 66 and set height to 100
+ Image prayutL	Store the boss prayut left side image
+ Image prayutR	Store the boss prayut right side image
+ Image prawitL	Store the boss prawit left side image
+ Image prawitR	Store the boss prawit right side image
+ Image podiumL	Store the podium left side image
+ Image podiumR	Store the podium right side image
+ Image status bar	Store the status bar image
+ Image heal	Store the heal item image, set width to 40 and set height to 40
+ Image attack	Store the attack item image, set width to 40 and set height to 40
+ Image cc	Store the cc item image, set width to 40 and set height to 40
+ Image revive	Store the revive item image, set width to 40 and set height to 40
+ Image immune	Store the immune item image, set width to 60 and set height to 60

+ Image immuneEffect	Store the immune effect image, set width to 110 and set height to 110
+ Image[] normalAttackEffect	Store images of normal attacking effect
+ Image[] healEffect	Store images of healing effect
+ Image[] stunEffect	Store images of stunned effect
+ Image[] slowEffect	Store images of slowed effect
+ Image[] silenceEffect	Store images of silenced effect
+ Image[] reviveEffect	Store images of reviving effect

2.2.2. Static Block

Set normalAttackEffect, healEffect, stunEffect, slowEffect, silenceEffect, reviveEffect image to array of each variables with ordered.

2.3. Class Sounds

This class contains all sounds.

2.3.1. Fields

+ AudioClip startbgm	Background music of start scene
+ AudioClip bgm	Background music of game scene
+ AudioClip deadscenebgm	Background music of dead scene
+ AudioClip attacksound	Normal attack sound effect
+ AudioClip deadsound	Sound of npc death

+ AudioClip healsound	Healing sound effect
+ AudioClip revivesound	Revive sound effect
+ AudioClip ccsound	CC item sound effect
+ AudioClip immunesound	Immune item sound effect
+ AudioClip spawnbosssound	Spawn boss sound effect

2.3.2. Static Block

Set game scene background music volume to 1%.
Set normal attack sound effect to 10%.

3. Package controller

3.1. Class GameManager

3.1.1. Fields

- GameManager instance = new GameManager()	Static object of GameManager
- boolean isGameRunning	True if the game is running
- List<Map> maps	List of all maps
- Player player	Game's player

3.2. Class MonsterAi

This class is a subclass of Thread class. It does monster generating job.

3.2.1. Constructor

+ MonsterGen()	Initialize and start a thread which randomly generates NPC in the current map every 1400 milliseconds using spawnMonsterRandom() method in Map class.

3.3. Class MonsterGen

This class is a subclass of Thread class. It does monster artificial intelligence job.

3.3.1. Constructor

+ MonsterAi()	Initialize and start a thread which apply NPC walking direction towards player and let the monster walk that way every 1/60 second using update() method in NPC class.

4. Package exception

This package contains all exceptions.

4.1. CannotAttackException

This exception is thrown when Character cannot attack.

4.2. CannotMoveException

This exception is thrown when Character cannot move.

4.3. CannotUseItemException

This exception is thrown when Character cannot use item.

4.4. InventoryFullException

This exception is thrown when Player’s inventory is full.

4.5. InventoryEmptyIndexException

This exception is thrown when Player tries to use item that player does not have.

4.6. ItemTypeNoExistException

This exception is thrown when Player tries to use item that type does not exist in the game.

5. Package input

5.1. Class KeyInput

This class contains all input keys available in the game.

5.1.1. Fields

- Set<KeyCode> activeKeys	A set containing all the keys that the user is pressing
- Queue<KeyCode> triggerkeys	A queue of keys which will be polled and processed by Player class
- final Set<KeyCode> UNPOLLABLE_KEYS	A set of keys that don't need to be in triggerKeys variable

5.1.3. Methods

Add some keyboard keys (Left, Right, Down, Up and Space) to UNPOLLABLE_KEYS

5.1.3. Methods

- void addKey(KeyCode code)	Add key to activeKeys and sometimes triggerKeys
- void removeKey(Keycode code)	Remove key from activeKeys
+ boolean pressingKey(KeyCode code)	Check whether the user is pressing the specific key
+ void clear()	Remove all keys in the activeKeys set
+ boolean isPollAvailable()	Check whether there is a content inside triggerKeys queue

+ KeyCode pollKey()	Poll key from triggerKeys
+ void bindScene(Scene scene)	Bind KeyInput methods to the scene

6. Package model

6.1. Interface IUpdatable

This interface is for entities that need to be updated.

6.1.1. Methods

+ void update()	Abstract method by default, update the entities

6.2. Interface IUseable

This interface is for entities that can be used.

6.2.1. Methods

+ void use()	Abstract method by default, use the entities that can be used.

6.3. Interface IMoveable

This interface is for entities that can be moved.

6.3.1. Methods

+ void move()	Abstract method by default, move the entities that can be moved.

6.4. Class Frame

This class is frames of every entities in the game.

6.4.1. Fields

# double posX	x-coordinate of the frame, start from left to right

# double posY	y-coordinate of the frame, start from top to bottom
# double width	Width of the frame
# double height	Height of the frame

6.4.2. Constructor

+ Frame(double posX, double poxY, double width, double height)	Initialize every fields.

6.3.3. Methods

+ boolean isCollideWith(Frame f)	Return true if this frame collides with frame f, otherwise false.
Getters & Setters of every fields	

6.5. Class Entity

This class represents every entities in the game.

6.5.1. Fields

# String name	Entity’s name
- Image image	Entity’s image

6.5.2. Constructor

+ Entity(double posX, double posY, double width, double height, String name, Image image)	Initialize every fields.

6.5.3. Methods

+ void render(GraphicContext gc)	Render the entity at coordinate (posX, posY) relative to the screen.
Getters of every fields & setter of image	

6.6. Class MoveableEntity

This class represents every moveable entities of the game.

6.6.1. Fields

+ <u>final int LEFT = -1</u>	Static values that represent the entity's facing direction
+ <u>final int RIGHT = 1</u>	
# double speedX	Speed in x-axis
# double speedY	Speed in y-axis
# int facing	Entity's facing direction (LEFT or RIGHT)
- Image imageL	Image when entity is facing left
- Image imageR	Image when entity is facing right

6.6.2. Constructors

+ MovableEntity(double posX, double posY, String name, Image imageL, Image imageR)	Initialize every fields, with width and height of the object equal to the width and height of imageL respectively, image is null, and is facing right.

6.6.3. Methods

+ void render(GraphicContext gc)	Render the entity at coordinate (posX, posY) relative to the screen.
+ void move()	Move the entity.
+ void setFacing(int facing)	Set facing direction of the entity, and change the image that corresponds to the direction
Getters & Setters of every fields	

6.7. Class Character

This class represents every characters of the game, including player and NPCs.

6.7.1. Fields

# int hp	Character’s health point
# int maxHp	Character’s maximum health point
# int minAtk	Character’s minimum attack point
# int maxAtk	Character’s maximum attack point
# int def	Character’s defense point
# CCType status	CC effect applied to the character (default is NONE)
# HpBar hpBar	Character’s health bar
- int animationTick	Frame counter for rendering animation (start from 0).
- int attackTick	Attack frame counter count since last attack frame (start from 0).
- int attackCooldown	The number of frame for character to be able to attack again.
- int ccedTick	CC frame counter since last CC applied frame (start from 0).
- int ccedDuration	The number of frames which CC is applied
- boolean isReviveAnimating	True if rendering revive animation (default is false)
- boolean isHealAnimating	True if rendering heal animation (default is false)
- boolean isAttacking	True if the attack is cooling down, otherwise false.

6.7.2. Constructors

+ Character(double posX, double posY, String name, Image imageL, Image imageR, int maxHp, int atk, int def, int attackCooldown)	Initialize every fields.

6.7.3. Methods

+ boolean isDead()	Return true if the character is dead, otherwise false.
+ boolean canAttack()	Return true if the character can attack, otherwise false.
+ boolean canMove()	Return true if the character can move, otherwise false.
+ boolean isCCed()	Return true if the character was applied CC effect, otherwise false.
+ boolean isStunned()	Return true if the character is stunned, otherwise false.
+ boolean isSlowed()	Return true if the character is slowed, otherwise false.
+ boolean isSilenced()	Return true if the character is silenced, otherwise false.
+ int getDamage()	Return random damage between minAtk and maxAtk
+ boolean takeDamage(int damage)	Called when the character is damaged. The damage is reduced by defense point. Return true if the character took damage (damage is more than 0), otherwise false.
+ void resetAttackTick()	Reset attackTick back to 0.
+ void addAttackTick()	Increase attackTick by 1. If attackTick equals to attackCooldown, reset attackTick and allow the character to attack.
+ void resetCCedTick()	Reset ccedTick back to 0.

+ void addCCedTick()	Increase ccedTick by 1. If ccedTick equals to ccedDuration, reset ccedTick and reset CC effect of the character.
+ void resetAnimationTick()	Reset animationTick back to 0.
+ void addAnimationTick()	Increase animationTick by 1. If animationTick equals to 59, reset animationTick and reset all animations.
+ void renderNormalAttack(GraphicContext gc)	Render normal attack animation.
+ void renderStatusEffect(GraphicContext gc)	Render CC effect of character.
+ void renderHealEffect(GraphicContext gc)	Render heal effect.
+ void renderReviveEffect(GraphicContext gc)	Render revive effect.
+ *void attack()*	*Throws CannotAttackException*
+ *void dead()*	Abstract method
Getters of hp, maxHp, isAttacking, isReviveAnimating, and isHealAnimating & setters of status, isAttacking, isReviveAnimating, isHealAnimating, and ccedDuration	

7. Package model.item

7.1. Class Item

7.1.1. Fields

- int count	Item counter (start from 0)

- int maxCount	Maximum number of item that can be collected
- final int expireTime = 300	The expire frame count since dropped
- int expireTick	Expire frame counter since dropped

7.1.2. Constructors

+ Item(String name, int maxCount, Image image)	Initialize every fields with posX = 0 and posY = 0.
+ Item(String name, int maxCount, Image image, double posX, double posY)	Initialize every fields.

7.1.3. Methods

+ boolean isExpired()	Return true if the item is iexpired, otherwise false.
+ addCount(int count)	Add the number of the item by the parameter if not exceeding maxCount. Return true if successfully add count, otherwise false.
+ void addExpireTick()	Increase expireTick by 1.
+ void update()	Update expireTick.
+ void use()	Activate the item if exists. Reduce count if activated successfully.
+ *boolean activate()*	*Activate the item, return true if activated successfully, otherwise false.*
+ void render(GraphicContext gc)	Render the item.
Getter of count	

7.2. Class AttackItem

7.2.1. Fields

- final int minDamage = 75	Minimum damage of item

- final int maxDamage = 150	Maximum damage of item

7.2.2. Constructors

+ AttackItem()	Initialize every fields with following value: - name = “Attack Item” - maxCount = 5 - image = Images.attack - posX = 0 - posY = 0
+ AttackItem(double posX, double posY)	Initialize every fields with following value: - name = “Attack Item” - maxCount = 5 - image = Images.attack

7.2.3. Methods

- int getDamage()	Return random damage of item between minDamage and maxDamage
+ boolean activate()	Damage all NPCs in the scene by the amount of getDamage(). Always return true.

7.3. Class CCItem

7.3.1. Constructors

+ CCItem()	Initialize every fields with following value: - name = “Random CC Item” - maxCount = 5 - image = Images.cc - posX = 0 - posY = 0

+ CCItem(double posX, double posY)	Initialize every fields with following value: - name = “Random CC Item” - maxCount = 5 - image = Images.cc

7.3.2. Methods

+ boolean activate()	Apply random CC effect between STUN, SLOW, and SILENCE to all NPCs in the scene. Always return true.

7.4. Class HealItem

7.4.1. Fields

- final int minHealHp = 200	Minimum healing HP
- final int maxHealHp = 500	Maximum healing HP

7.4.2. Constructors

+ HealItem()	Initialize every fields with following value: - name = “Heal Potion” - maxCount = 5 - image = Images.heal - posX = 0 - posY = 0
+ HealItem(double posX, double posY)	Initialize every fields with following value: - name = “Heal Potion” - maxCount = 5 - image = Images.heal

7.4.3. Methods

- int getHealHp()	Return random healing HP of item between minHealHp and maxHealHp

+ boolean activate()	Heal the player by the amount of getHealHp(). Return false if player's HP is max.

7.5. Class ImmuneItem

7.5.1. Fields

+ <u>final int duration = 100</u>	Duration of immune effect

7.5.2. Constructors

+ ImmuneItem()	Initialize every fields with following value: - name = "Immune Item" - maxCount = 5 - image = Images.immune - posX = 0 - posY = 0
+ ImmuneItem(double posX, double posY)	Initialize every fields with following value: - name = "Immune Item" - maxCount = 5 - image = Images.immune

7.5.3. Methods

+ boolean activate()	Make player immune to damage while in duration. Return false if player has already immune.

7.6. Class ReviveItem

7.6.1. Constructors

+ ReviveItem()	Initialize every fields with following value: - name = “Revive Item” - maxCount = 1 - image = Images.revive - posX = 0 - posY = 0
+ ReviveItem(double posX, double posY)	Initialize every fields with following value: - name = “Revive Item” - maxCount = 1 - image = Images.revive

7.6.2. Methods

+ boolean activate()	Make player revivable. Return false if player has been already revivable.

8. Package model.map

8.1. Class Map

8.1.1. Fields

- final int X_PADDING = 150	Both axes padding
- final int Y_PADDING = 66	
- Image image	Map’s image
- double moveSpeed	Map’s move speed (default is 4)
- MediaPlayer bgm	Background music player
- List<NPC> listNPC	List of all NPCs in the map
- List<Item> listItem	List of all items in the map
- List<Boss> listBoss	List of all bosses in the map
- List<Podium> listPodium	List of all podium effect in the map
- StatusBar statusBar	Status bar of player

8.1.2. Constructor

+ Map(Image image, AudioClip bgm)	Initialize map with background image and bgm.

8.1.3. Methods

+ List<NPC> collideNPCs(Frame f)	Return list of all NPCs that collide with Frame f.
+ List<Item> collideItems(Frame f)	Return list of all items that collide with Frame f.
+ void addItem(Item item)	Add item to the map.
+ void removeItem(Item item)	Remove item from the map.
+ void addPodium(Podium podium)	Add podium to the map.
+ void removePodium(Podium podium)	Remove podium from the map.
- void moveMap()	Move the map when player tries to move out of screen.
- void motion(MoveableEntity e)	Move the MoveableEntity e in the map, except player.
+ void motionPlayer(Player p)	Move player and map so the player is always on screen.
+ void motionAll()	Move all MoveableEntity in the map except player.
+ void spawnMonsterRandom()	Spawn an NPC with random coordinate.
+ void spawnBossRandom()	Spawn a boss with random coordinate.
+ void rander(GraphicContext gc)	Render everything in the map.
+ void update()	Update every updatable entities in the map.
+ void playBgm()	Play BGM sound.
Getters of listNPC, listItem, listBoss, and listPodium	

9. Package model.player

9.1. Class Player

9.1.1. Fields

- Item[] inventory	Player’s inventory, consists of [AttackItem, CCItem, HealItem, ImmuneItem]
- boolean isImmune	True if player is immune (default false)
- boolean isRevivable	True if player can be revived (default false)
- int immuneTick	Frame counter of player’s immunity (start from 0)
- double attackRange	Player’s attack range (default 30)

9.1.2. Constructor

+ Player(double posX, double posY)	Initialize player with following values: - posX, posY from parameters - name = “Netikun” - imageL = Images.playerL - imageR = Images.playerR - maxHp = 3000 - minAtk = 50 - maxAtk = 150 - def = 50 - attackCooldown = 30

9.1.3. Methods

+ boolean isDead()	Return true if player is dead and cannot be revived.
+ boolean canUseItem()	Return true if player is alive and no CC applied.
+ boolean heal(int hp)	Heal player by hp. Return false if health has been already max.
+ void refresh()	Refresh hp to maxHp.

+ void revive()	Revive player and clear all CC effect.
+ void attack()	Throws CannotAttackException when player cannot attack. Deal damage and knock back to every NPCs within attack area.
+ boolean takeDamage(int damage)	Called when player is damaged. The damage is reduced by defense point. Return false if player did not take damage (damage is 0 or player is immune).
+ void dead()	Called when player is dead. Stop player motion and play death scene.
+ Frame getPlayerArea()	Return player’s frame.
+ Frame getAttackArea()	Return player’s attack area frame.
+ void resetImmuneTick()	Reset immune

10. Package model.npc

10.1. Class NPC

10.1.1. Fields

- double speed	Random speed for NPC

10.1.2. Constructor

+ NPC(double posX, double posY)	Initialize player with following values: - posX, posY from parameters - name = “Soldier” - imageL = Images.soldierL - imageR = Images.soldierR - maxHp = 100 - minAtk = 50 - maxAtk = 150 - def = 50 - attackCooldown = 120 - speed = random value between 1 and 2
+ NPC(double posX, double posY, String name, Image imageL, Image imageR, int maxHp, int minAtk, int maxAtk, int def, int attackCooldown)	Initialize every fields with parameters and set speed to 0.

10.1.3. Methods

+ void attack()	Throws CannotAttackException if cannot attack. Automatically attacks with random damage when collides with player.
+ void dead()	Called when NPC is dead.
+ void dropItem()	Drop item when died.
+ void update()	Update NPC’s parameters for each frame.
+ void render(GraphicContext gc)	Render NPC and HP bar.
Setter of speed	

10.2. Class Boss

10.2.1. Fields

- Skill skill	Boss’ skill

10.2.2. Constructor

+ Boss(double posX, double posY, String name, Image imageL, Image imageR, int maxHp, int minAtk, int maxAtk, int def, Skill skill)	Initialize every fields with attackCooldown is 60.

10.2.3. Methods

+ void attack()	Similar to NPC’s attack with additional skill activation.
+ void update()	Similar to NPC’s update with additional skill update.
Getter of skill	

10.3. Class Prawit

This class is Prawit boss model.

10.3.1. Constructor

+ Prawit(double posX, double posY)	Initialize every fields with following values: - posX, posY from parameters - name = “Prawit” - imageL = Images.prawitL - imageR = Images.prawitR - maxHp = 5000 - minAtk = 50 - maxAtk = 200 - def = 80 - skill = WatchSkill object
+ void dead()	Generate Prayut boss when died.

10.4. Class Prayut

This class is Prayut boss model.

10.4.1. Constructor

+ Prayut(double posX, double posY)	Initialize every fields with following values: - posX, posY from parameters - name = “Prayut” - imageL = Images.prayutL - imageR = Images.prayutR - maxHp = 10000 - minAtk = 100 - maxAtk = 500 - def = 100 - skill = PodiumSkill object
+ void dead()	Generate Prawit boss when died.

11. Package model.effect

11.1. Class Podium

This class is podium effect for PodiumSkill

11.1.1. Constructor

+ Podium(double posX, double posY)	Initialize every fields with following values: - posX, posY from parameters - imageL = Images.podiumL - imageR = Images.podiumR - speedX = random value between 2 - 7 - speedY = 0 - facing = toward player

11.1.2. Methods

+ boolean isOutOfWindow()	Return true if the podium is out of window.
+ void update()	Update podium state every frame.

12. Package skill

12.1. Class Skill

12.1.1. Fields

- int cooldownTime	Skill cooldown time
- int cooldownTick	Skill cooldown frame count since last used (start from 0)
- boolean isCoolingDown	True if skill is cooling down (default is false)

12.1.2. Constructor

+ Skill(int cooldownTime)	Initialize every fields

12.1.3. Methods

+ void resetCooldownTick()	Reset cooldownTick back to 0.
+ void addCooldownTick()	Increase cooldownTick by 1. If it equals to cooldownTIme. set cooling to false and reset cooldownTick.
+ void update()	Call addCooldownTick() in every frame.
Getter & Setter of isCoolingDown	

12.2. Class PodiumSkill

12.2.1. Constructor

+ PodiumSkill()	Set skill cooldown to 240

12.2.2. Method

+ void use()	Throw a podium with random y-coordinate from either left or right corner of the screen.

12.3. Class WatchSkill

12.2.1. Constructor

+ WatchSkill()	Set skill cooldown to 600

12.2.2. Method

+ void use()	Throw a podium with random y-coordinate from either left or right corner of the screen.

13. Package ui

13.1. Class StartScene

13.1.1. Fields

<u>+ final double WINDOW_WIDTH = 420</u>	Start scene window dimension
<u>+ final double WINDOW_HEIGHT = 540</u>	
- Pane root	Store pane of the scene graph
- Canvas canvas	Canvas for draw start scene
- MediaPlayer bgm	Background music player

13.1.2. Constructor

+ StartScene()	Initialize start scene

13.1.3. Methods

- boolean isOnPlayButton(MouseEvent event)	Return true of mouse cursor is on play button.
- boolean isOnInstructionsButton(MouseEvent event)	Return true of mouse cursor is on instructions button.
- void addCanvasEventHandler()	Add event handlers in game canvas.

13.2. Class GameScene

13.2.1. Fields

+ final double WINDOW_WIDTH = 900	Game window dimension
+ final double WINDOW_HEIGHT = 600	
- Canvas canvas	Canvas for draw game scene

13.2.2. Constructor

+ GameScene()	Initialize game scene

13.2.3. Methods

- boolean isOnQuitButton(MouseEvent event)	Return true of mouse cursor is on quit button.
- void addCanvasEventHandler()	Add event handlers in game canvas.

13.3. Class InstructionsScene

13.3.1. Fields

- Pane root	Store pane of the scene graph
- Canvas canvas	Canvas for draw instruction scene
- Image[] instructions	Instruction images array
- int count	Page count (start from 0)

13.3.2. Constructor

+ InstructionsScene()	Initialize instructions scene

13.3.3. Methods

- boolean isOnNextButton(MouseEvent event)	Return true of mouse cursor is on next button.
- boolean isOnPrevButton(MouseEvent event)	Return true of mouse cursor is on prev button.
- void addCanvasEventHandler()	Add event handlers in game canvas.

13.4. Class StatusBar

13.4.1. Fields

+ final double WINDOW_HEGIHT = 600	Window height
+ final double HEIGHT = 50	Status bar height
+ final double HP_WIDTH = 200	Player’s HP bar dimension
+ final double HP_HEIGHT = 16	
+ final double HP_X = 100	Player’s bar coordinate
+ final double HP_Y = 10	
- final Color HP_COLOR = Color.GREEN.brighter()	HP bar’s color
- final Font HP_BAR_FONT	Hp bar’s font, which is Tahoma Bold 12
- final Font ITEM_FONT	Item count font, which is Tahoma Bold 15
- final Font TIME_FONT	Time font, which is Tahoma Bold 20
- double hpWidth	Width of player’s HP in HP bar (green section, default 0)
- Image img = Images.statusbar	Status bar image

13.4.2. Method

+ render(GraphicContext gc)	Render player’s status bar, including HP bar and inventory.

13.5. Class HpBar

13.5.1. Fields

+ final double WIDTH = 60	HP bar dimension
+ final double HEIGHT = 6	
- Character character	Owner of HP bar

13.5.2. Constructor

+ HpBar(Character character)	Assign character to HP bar

13.5.3. Method

+ render(GraphicContext gc)	Render HP bar below the character. If this character is player and has revive item render revive icon next to HP bar.