



โครงการ

อัศวินแห่งความมืด

(Dark Knight)

จัดทำโดย

6604062636151 นายณัฐชานน ทรัพย์มีชัย

เสนอ

ผู้ช่วยศาสตราจารย์สถิตย์ ประสมพันธ์

รายงานนี้เป็นส่วนหนึ่งของวิชา 040613204 การโปรแกรมเชิงวัตถุ

(Object-oriented Programming)

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์ มหาวิทยาลัย

เทคโนโลยีพระจอมเกล้าพระนครเหนือ

ภาคเรียนที่ 1 ปีการศึกษา 2567

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงการ

โครงการนี้จัดขึ้นเพื่อวัดความสามารถในการเรียนวิชา Object Oriented Programming โดยการนำเรื่องที่เรียนมาสร้างชิ้นงานในรูปแบบของเกม โดยใช้แนวคิดการเขียนโปรแกรมแบบเชิงวัตถุเพื่อช่วยในการแก้ปัญหาต่างๆภายในเกม

1.2 ประเภทของโครงการ

- เกม (Game)

1.3 ประโยชน์

- 1.เพื่อความสนุกสนาน
- 2.ฝึกทักษะการวางแผนและการแก้ปัญหา
- 3.พัฒนาทักษะการคิดและการตัดสินใจ
- 4.การพัฒนาปฏิกิริยาตอบสนอง

1.4 ขอบเขตของโครงการ

ตารางเวลาการดำเนินโครงการ

| ลำดับ | รายการ | 1-5 ต.ค. | 6-25 ต.ค. | 26-31 ต.ค. |
|-------|-------------------------------------|----------|-----------|------------|
| 1 | หารูปตัวละคร แมพ และ ทำกราฟฟิกต่างๆ | | | |
| 2 | ศึกษาเอกสารและข้อมูลที่เกี่ยวข้อง | | | |
| 3 | ลงมือเขียนโปรแกรม | | | |
| 4 | จัดทำเอกสาร | | | |
| 5 | ตรวจสอบและแก้ไขข้อผิดพลาด | | | |

บทที่ 2

ส่วนการพัฒนา

2.1 เนื้อเรื่องย่อ

เป็นเกมที่ผู้เล่นสวมบทบาทเป็นอัศวินผู้กล้าที่ได้รับอาวุธดาบตำนานจากพระเจ้าเพื่อมอบหมายให้ไปปราบปีศาจแห่งความมืดที่มารุกรานเหล่าผู้คนในเมืองมาอย่างยาวนานเพื่อที่จะให้เมืองของพวกเขากลับมา สงบสุขอีกครั้งโดยแต่ละปีศาจจะมีธาตุเป็นของตัวเองต้องใช้ดาบธาตุนั้นในการกำจัด และตัวละครจะมีหลอด HP แสดงสถานะเลือดในเกม และ เกมจะจบลงก็ต่อเมื่อผู้เล่นนั้น HP หมด หรือ ผู้เล่นนั้นสามารถพิชิตปีศาจ แห่งความมืดได้สำเร็จ

2.2 วิธีการเล่น

ผู้เล่นจะต้องใช้ความสามารถต่างๆเพื่อกำจัดMonsterต่างๆเพื่อให้ผ่านด่านจนไปถึงBoss และทำการกำจัดBoss และ Monster ให้หมดถึงจะชนะระหว่างทางมี Item Drop เพื่อช่วยผู้เล่นในการเอาชีวิตรอดและแต่ละ Monster จะมีธาตุเป็นของตัวเองต้องใช้ดาบธาตุนั้นในการกำจัด

| | |
|--|---|
| <div>เลือกเมนู</div>  | <div>-กด Start เพื่อเล่นเกม</div> <div>-กด EXIT เพื่อออกเกม</div> |
| <div>เลือกด่าน</div>  | <div>-เลือกด่านที่ต้องการ</div> |

รายละเอียดต่างๆ



-หลอด AP

ใช้บอกว่าตัวละครมีเกราะ (Armor) ตอนนั้นเท่าไรถ้าเกราะหมดจะไปลดที่เลือดแทน

-หลอด HP

ใช้บอกว่าตัวละครมีพลังชีวิตตอนนั้นมีเท่าไรถ้าหมดเกมจะจบและจะแพ้ในเกมนั้น

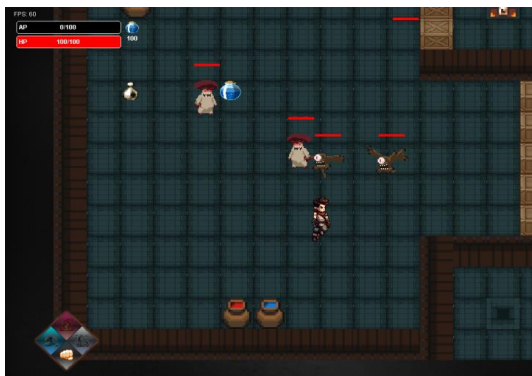
-มานา (Mana)

ใช้บอกว่าตัวละครมีManaตอนนั้นเท่าไรถ้า Mana หมดจะไม่สามารถใช้ดาบได้

-ธาตุ (Wheel of elements)

ใช้บอกว่าในตอนนั้นใช้ธาตุอะไรอยู่

การเดิน



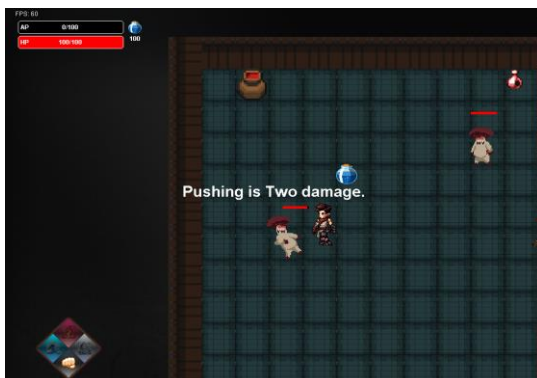
-กด w เพื่อเดินข้างบน

-กด a เพื่อเดินไปทางซ้าย

-กด s เพื่อเดินทางข้างล่าง


-กด d เพื่อเดินไปทางขวา

หมัด (Punch)

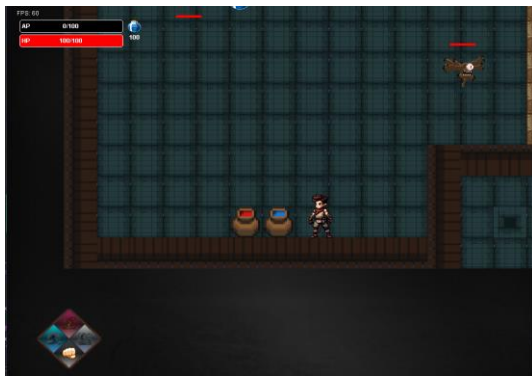


-กด 1 เพื่อกลับเป็นหมัด (Punch)

-กด space bar เพื่อโจมตี

| | |
|--|--|
| <p>ธาดุไฟ</p>  | <p>-กดเลข 2 เพื่อใช้ดาบธาดุไฟ</p> <p>-กด space bar เพื่อโจมตี</p> <p>ใช้โจมตีได้แค่ Bringer of Death (Boss)</p> <p>(ธาดุไฟ)</p> |
| <p>ธาดุน้ำ</p>  | <p>-กดเลข 3 เพื่อใช้ดาบธาดุน้ำ</p> <p>-กดspace bar เพื่อโจมตี</p> <p>ใช้โจมตีได้แค่ Mushroom (ธาดุน้ำ)</p> |
| <p>ธาดุลม</p>  | <p>-กดเลข 4 เพื่อใช้ดาบธาดุลม</p> <p>-กดspace bar เพื่อโจมตี</p> <p>ใช้โจมตีได้แค่ Flyingeye (ธาดุลม)</p> |
| <p>Item Drop</p>  | <ol style="list-style-type: none"> 1.Item chest ช่วยเพิ่มความแรงดาบ + 5 2.Item HP ช่วยเพิ่ม HP +50 3.Item Armor ช่วยเพิ่ม AP +50 4.Item Mana ช่วยเพิ่ม Mana + 50 5.Item BootSpeed ช่วยเพิ่ม ความเร็วในการเดิน + 5 ใช้ได้ 5 วินาที |
| | |

Object Drop



1.VeganHealth

ถ้าโจมตีทำลายให้แตกได้ HP + 30

2.VeganMana

ถ้าโจมตีทำลายให้แตกได้ Mana + 30

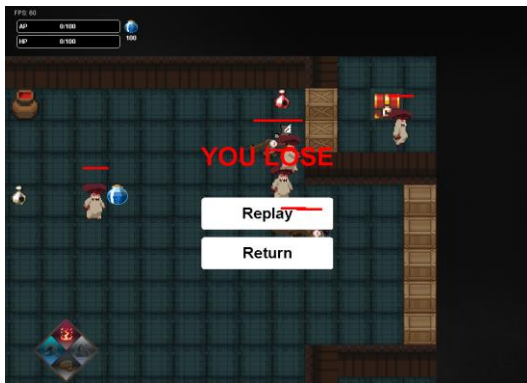
Winner



1.Replay เริ่มเล่นใหม่ด่านเดิมอีกครั้ง

2.Return กลับไปหน้าหลัก

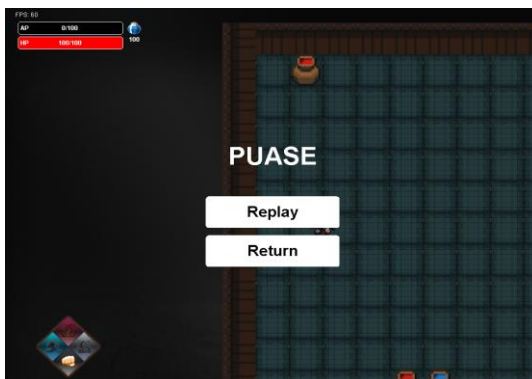
LOSE



1.Replay เริ่มเล่นใหม่ด่านเดิมอีกครั้ง

2.Return กลับไปหน้าหลัก

PUASE



-กด ESC เพื่อเปิด / ปิด

1.Replay เริ่มเล่นใหม่ด่านเดิมอีกครั้ง

2.Return กลับไปหน้าหลัก

2.3 Class diagram

| Game |
|---------------------|
| Game() |
| main(String[]) void |

| GamePanel |
|-------------------------------------|
| GamePanel() |
| objects ArrayList<Objects> |
| monster ArrayList<Entity> |
| playerState int |
| gameWinState int |
| assetSetterObject AssetSetterObject |
| titleSize int |
| player Player |
| bgGame BufferedImage |
| keyH KeyHandler |
| maxScreenCol int |
| UIStatus UIStatus |
| map Supermap |
| FPS int |
| npc ArrayList<Entity> |
| screenHeight int |
| ui UI |
| maxScreenRow int |
| FrameRate int |
| selectMapState int |
| scale int |
| menuState int |
| timerGame Timer |
| screenWidth int |
| gameState int |
| gameThread Thread |
| currentTime int |
| pauseState int |
| gameOverState int |
| originalTitleSize int |
| cChecker CollisionChecker |
| paintComponent(Graphics) void |
| startGameThread() void |
| setupGame() void |
| DrawFPS(Graphics) void |
| run() void |
| update() void |
| resetStateAll() void |
| actionPerformed(ActionEvent) void |
| setBackgroundGame() void |

| Entity |
|--|
| Entity(GamePanel) |
| right ArrayList<BufferedImage> |
| spriteNum int |
| worldY int |
| worldX int |
| down ArrayList<BufferedImage> |
| isMoving boolean |
| speed int |
| spriteCounter int |
| solidAreaDefaultY int |
| imageHeight int |
| up ArrayList<BufferedImage> |
| direction String |
| left ArrayList<BufferedImage> |
| gp GamePanel |
| solidArea Rectangle |
| attackArea Rectangle |
| imageWidth int |
| lastDirection String |
| collisionOn boolean |
| solidAreaDefaultX int |
| setAction() void |
| draw(Graphics) void |
| getImageHeight() int |
| getEntityCoords() HashMap<String, Integer> |
| getImageWidth() int |
| setImageHeight(int) void |
| loadAnimation() void |
| setImageWidth(int) void |
| update() void |
| loadSprite(String) BufferedImage? |

| Texture |
|----------------------------------|
| Texture(int) |
| dict Map<Integer, TextureLoader> |
| GRASS_2 |
| WALL_BOTTOM |
| GRASS_3 |
| GRASS_1 |
| STONE |
| WALL_LEFT |
| GRASS_4 |
| FLOOR |
| grassWall |
| EMPTY |
| GRASS_STONE1 |
| FLOOR_1 |
| GRASS_STONE2 |
| WALL_TOP |
| WALL_RIGHT |
| BoxWood1 |
| valueOf(String) Texture |
| setCollision(boolean) void |
| loadTexture() void |
| getTexture() BufferedImage |
| getTextureId() int |
| isCollision() boolean |
| getTextureWidth() int |
| values() Texture[] |
| getTextureHeight() int |

| ObjectsEnum |
|-----------------------------|
| ObjectsEnum(int) |
| POLE |
| Thorn |
| VeganHealth |
| GODDESS |
| BootSpeed |
| VeganMana |
| chest |
| BoxWood |
| tree |
| objectId int |
| SPAWN |
| armor |
| BoxDoor |
| Health |
| dict Map<Integer, Objects> |
| mana |
| copy() Objects |
| values() ObjectsEnum[] |
| valueOf(String) ObjectsEnum |
| loadObjects() void |
| getObjectId() int |

| CollisionChecker |
|--|
| CollisionChecker(GamePanel) |
| gp GamePanel |
| checkPlayerAttackObject(Entity, ArrayList<Objects>) ArrayList<Integer> |
| checkPlayerAttackMonster(Entity, ArrayList<Entity>) ArrayList<Integer> |
| checkObject(Entity, boolean) int |
| checkMap(Entity) void |
| checkEntity(Entity, ArrayList<Entity>) int |
| checkPlayer(Entity) boolean |

| Element |
|---------------------------|
| Element(String, int, int) |
| damage int |
| manaCost int |
| image BufferedImage |
| setDamage(int) void |

| ElementEnums |
|--------------------------------------|
| ElementEnums(int) |
| FIRE |
| WIND |
| PUNCH |
| WATER |
| dict Map<Integer, Element> |
| elementId int |
| valueOf(String) ElementEnums |
| setDamageElementId(int, int) void |
| values() ElementEnums[] |
| loadElements() void |
| getImageElementId(int) BufferedImage |
| getDamageElementId(int) int |
| getElementId() int |
| getManaCostElementId(int) int |

| TextureLoader |
|--------------------------------|
| TextureLoader(String, boolean) |
| collision boolean |
| image BufferedImage |
| imageHeight int |
| imageWidth int |
| setCollision(boolean) void |
| getTextureWidth() int |
| getImage() BufferedImage |
| isCollision() boolean |
| getTextureHeight() int |

| UI |
|---------------------------------|
| UI(GamePanel) |
| buttonStart Image |
| buttonExit Image |
| gp GamePanel |
| bgMenu Image |
| bg Image |
| g2 Graphics2D |
| commandNum int |
| mouseExited(MouseEvent) void |
| gameWin() void |
| getXforCenteredText(String) int |
| gameOver() void |
| selectMap() void |
| pauseMenu() void |
| menuStartGame() void |
| mousePressed(MouseEvent) void |
| mouseClicked(MouseEvent) void |
| draw(Graphics2D) void |
| mouseReleased(MouseEvent) void |
| mouseEntered(MouseEvent) void |

| <div> <div></div> <div>UIStatus</div> </div> | |
|--|---------------|
| <div> <div></div> <div>UIStatus(GamePanel)</div> </div> | |
| <div> <div></div> <div>mana</div> </div> | BufferedImage |
| <div> <div></div> <div>gp</div> </div> | GamePanel |
| <div> <div></div> <div>cooldownAlert</div> </div> | int |
| <div> <div></div> <div>alertText</div> </div> | String |
| <div> <div></div> <div>g2</div> </div> | Graphics2D |
| <div> <div></div> <div>setAlert(String, int)</div> </div> | void |
| <div> <div></div> <div>drawAlert()</div> </div> | void |
| <div> <div></div> <div>draw(Graphics2D)</div> </div> | void |
| <div> <div></div> <div>drawCurrentPlayTime()</div> </div> | void |
| <div> <div></div> <div>drawBoxArmor()</div> </div> | void |
| <div> <div></div> <div>drawElement()</div> </div> | void |
| <div> <div></div> <div>drawCoords()</div> </div> | void |
| <div> <div></div> <div>drawBoxHealthMonster()</div> </div> | void |
| <div> <div></div> <div>drawMana()</div> </div> | void |
| <div> <div></div> <div>drawBoxHealth()</div> </div> | void |

| <div> <div></div> <div>AssetSetterObject</div> </div> | |
|---|-----------|
| <div> <div></div> <div>AssetSetterObject(GamePanel)</div> </div> | |
| <div> <div></div> <div>gp</div> </div> | GamePanel |
| <div> <div></div> <div>draw(Graphics)</div> </div> | void |
| <div> <div></div> <div>setSpawnObjects(int, int, int, int, int, int)</div> </div> | void |

| <div> <div></div> <div>Monster</div> </div> | |
|--|--------------------------|
| <div> <div></div> <div>Monster(GamePanel)</div> </div> | |
| <div> <div></div> <div>element</div> </div> | int |
| <div> <div></div> <div>ThreadDelay</div> </div> | int |
| <div> <div></div> <div>maxHealth</div> </div> | int |
| <div> <div></div> <div>rebound</div> </div> | int |
| <div> <div></div> <div>countHit</div> </div> | int |
| <div> <div></div> <div>MonsterThread</div> </div> | Thread |
| <div> <div></div> <div>health</div> </div> | int |
| <div> <div></div> <div>DestinationActionLockCounter</div> </div> | int |
| <div> <div></div> <div>actionLockCounter</div> </div> | int |
| <div> <div></div> <div>idle</div> </div> | ArrayList<BufferedImage> |
| <div> <div></div> <div>hit</div> </div> | ArrayList<BufferedImage> |
| <div> <div></div> <div>attackDamage</div> </div> | int |
| <div> <div></div> <div>attack</div> </div> | ArrayList<BufferedImage> |
| <div> <div></div> <div>AttacktoPlayer()</div> </div> | void |
| <div> <div></div> <div>getMaxHealth()</div> </div> | int |
| <div> <div></div> <div>draw(Graphics)</div> </div> | void |
| <div> <div></div> <div>AttackedByPlayer(int)</div> </div> | void |
| <div> <div></div> <div>isDead()</div> </div> | boolean |
| <div> <div></div> <div>setDefaultValues(int, int)</div> </div> | void |
| <div> <div></div> <div>stopMonsterThread()</div> </div> | void |
| <div> <div></div> <div>update()</div> </div> | void |
| <div> <div></div> <div>setAction()</div> </div> | void |
| <div> <div></div> <div>setHealth(int)</div> </div> | void |
| <div> <div></div> <div>loadAnimation()</div> </div> | void |
| <div> <div></div> <div>run()</div> </div> | void |
| <div> <div></div> <div>getHealth()</div> </div> | int |
| <div> <div></div> <div>startMonsterThread()</div> </div> | void |

| <div> <div></div> <div>PlayerState</div> </div> | |
|--|---------|
| <div> <div></div> <div>PlayerState()</div> </div> | |
| <div> <div></div> <div>health</div> </div> | int |
| <div> <div></div> <div>isAttacking</div> </div> | boolean |
| <div> <div></div> <div>currentElement</div> </div> | int |
| <div> <div></div> <div>itemDelay</div> </div> | int |
| <div> <div></div> <div>mana</div> </div> | int |
| <div> <div></div> <div>armor</div> </div> | int |
| <div> <div></div> <div>map</div> </div> | int[] |
| <div> <div></div> <div>countKilled</div> </div> | int |

| <div> <div></div> <div>Supermap</div> </div> | |
|---|-----------|
| <div> <div></div> <div>Supermap(GamePanel)</div> </div> | |
| <div> <div></div> <div>MapContentet</div> </div> | int[][] |
| <div> <div></div> <div>gp</div> </div> | GamePanel |
| <div> <div></div> <div>currentTimeMap</div> </div> | int |
| <div> <div></div> <div>timerMap</div> </div> | Timer |
| <div> <div></div> <div>countMonster</div> </div> | int |
| <div> <div></div> <div>currentMonster</div> </div> | int |
| <div> <div></div> <div>setDefaultObjects()</div> </div> | void |
| <div> <div></div> <div>resetCountKilled()</div> </div> | void |
| <div> <div></div> <div>actionPerformed(ActionEvent)</div> </div> | void |
| <div> <div></div> <div>update()</div> </div> | void |
| <div> <div></div> <div>removeBoxDoor()</div> </div> | void |
| <div> <div></div> <div>draw(Graphics)</div> </div> | void |
| <div> <div></div> <div>getMonsterCount()</div> </div> | int |
| <div> <div></div> <div>setDefaultSpawnEntityAndObjects()</div> </div> | void |

| <div> <div></div> <div>M1_ST1</div> </div> | |
|---|------|
| <div> <div></div> <div>M1_ST1(GamePanel)</div> </div> | |
| <div> <div></div> <div>setDefaultObjects()</div> </div> | void |
| <div> <div></div> <div>update()</div> </div> | void |

| <div> <div></div> <div>M1_ST2</div> </div> | |
|---|------|
| <div> <div></div> <div>M1_ST2(GamePanel)</div> </div> | |
| <div> <div></div> <div>setDefaultObjects()</div> </div> | void |
| <div> <div></div> <div>update()</div> </div> | void |

| <div> <div></div> <div>BringerOfDeath</div> </div> | |
|---|------|
| <div> <div></div> <div>BringerOfDeath(GamePanel, int, int)</div> </div> | |
| <div> <div></div> <div>loadAnimation()</div> </div> | void |
| <div> <div></div> <div>setDefaultValues(int, int)</div> </div> | void |
| <div> <div></div> <div>draw(Graphics)</div> </div> | void |

| <div> <div></div> <div>KeyHandler</div> </div> | |
|---|-----------|
| <div> <div></div> <div>KeyHandler(GamePanel)</div> </div> | |
| <div> <div></div> <div>gp</div> </div> | GamePanel |
| <div> <div></div> <div>left</div> </div> | boolean |
| <div> <div></div> <div>right</div> </div> | boolean |
| <div> <div></div> <div>down</div> </div> | boolean |
| <div> <div></div> <div>up</div> </div> | boolean |
| <div> <div></div> <div>keyTyped(KeyEvent)</div> </div> | void |
| <div> <div></div> <div>keyPressed(KeyEvent)</div> </div> | void |
| <div> <div></div> <div>keyReleased(KeyEvent)</div> </div> | void |

| <div> <div></div> <div>M2_ST1</div> </div> | |
|---|------|
| <div> <div></div> <div>M2_ST1(GamePanel)</div> </div> | |
| <div> <div></div> <div>setDefaultObjects()</div> </div> | void |
| <div> <div></div> <div>update()</div> </div> | void |

| <div> <div></div> <div>Mushroom</div> </div> | |
|---|------|
| <div> <div></div> <div>Mushroom(GamePanel, int, int)</div> </div> | |
| <div> <div></div> <div>setDefaultValues(int, int)</div> </div> | void |
| <div> <div></div> <div>loadAnimation()</div> </div> | void |
| <div> <div></div> <div>draw(Graphics)</div> </div> | void |

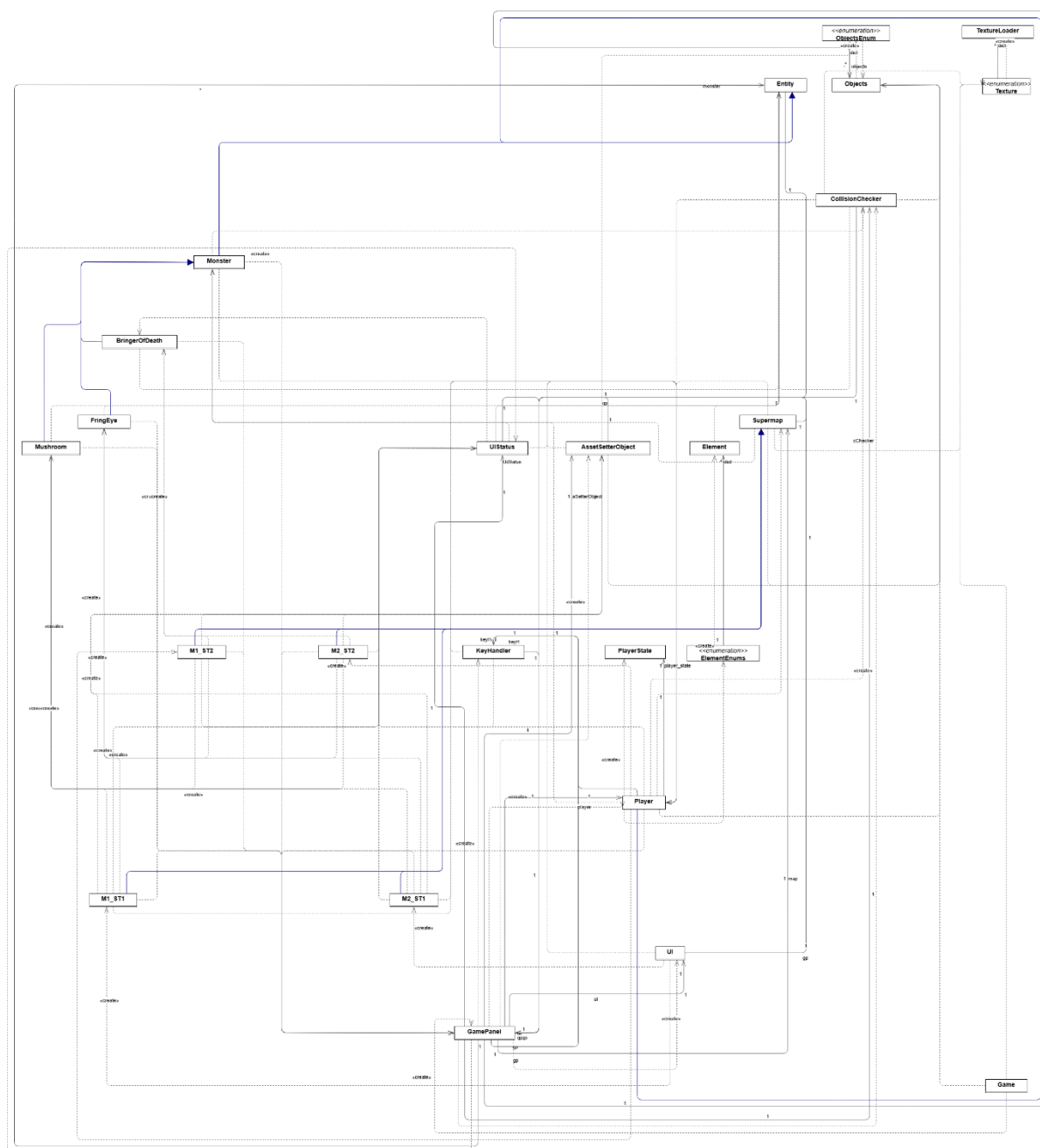
| <div> <div></div> <div>M2_ST2</div> </div> | |
|---|------|
| <div> <div></div> <div>M2_ST2(GamePanel)</div> </div> | |
| <div> <div></div> <div>setDefaultObjects()</div> </div> | void |
| <div> <div></div> <div>update()</div> </div> | void |

| <div> <div></div> <div>FringEye</div> </div> | |
|---|------|
| <div> <div></div> <div>FringEye(GamePanel, int, int)</div> </div> | |
| <div> <div></div> <div>draw(Graphics)</div> </div> | void |
| <div> <div></div> <div>loadAnimation()</div> </div> | void |
| <div> <div></div> <div>setDefaultValues(int, int)</div> </div> | void |

| Player | |
|-----------------------------------|--------------------------|
| Player(GamePanel, KeyHandler) | |
| keyH | KeyHandler |
| attack_wind | ArrayList<BufferedImage> |
| screenY | int |
| screenX | int |
| attack_fire | ArrayList<BufferedImage> |
| actionThornCounter | int |
| idle | ArrayList<BufferedImage> |
| attack_water | ArrayList<BufferedImage> |
| player_state | PlayerState |
| damageObject(ArrayList<Integer>) | void |
| InteractNPC(int) | void |
| setArmor(int) | void |
| setAction() | void |
| draw(Graphics) | void |
| loadAnimation() | void |
| getHealth() | int |
| setDefaultValues() | void |
| setCountKilled(int) | void |
| setItemDelay(int) | void |
| restoreMana() | void |
| getStateMap() | int[] |
| setCurrentElement(int) | void |
| getCurrentElement() | int |
| setMap(int, int) | void |
| setHealth(int) | void |
| checkBootSpeedItemDelay() | void |
| getArmor() | int |
| update() | void |
| getMana() | int |
| playerAttacked(int) | void |
| setMana(int) | void |
| checkEventObject(int) | void |
| checkPlayerStateMap() | void |
| isAttacking() | boolean |
| getCountKilled() | int |
| getCurrentManaCost() | int |
| getItemDelay() | int |
| damageMonster(ArrayList<Integer>) | void |
| isDead() | boolean |
| getImageCurrentElement() | BufferedImage |
| ActionAttack() | void |
| setAttacking(boolean) | void |

| Objects | |
|--|---------------|
| Objects(BufferedImage, boolean, int) | |
| Objects(String, boolean, boolean, int) | |
| image | BufferedImage |
| solidAreaDefaultY | int |
| solidArea | Rectangle |
| imageHeight | int |
| mapId | int[] |
| objectId | int |
| collision | boolean |
| health | int |
| imageWidth | int |
| worldX | int |
| isAttacking | boolean |
| worldY | int |
| solidAreaDefaultX | int |
| getSolidArea() | Rectangle |
| isAttacking() | boolean |
| setCollision(boolean) | void |
| setObjectHeight(int) | void |
| getObjectId() | int |
| setAttacking(boolean) | void |
| getObjectHeight() | int |
| getWorldY() | int |
| setWorldY(int) | void |
| getSolidAreaDefaultX() | int |
| ObjectIsDead() | boolean |
| setWorldX(int) | void |
| copy() | Objects |
| AttackedByPlayer(int) | void |
| setObjectWidth(int) | void |
| getWorldX() | int |
| getObjectWidth() | int |
| getHealth() | int |
| getObject() | BufferedImage |
| getSolidAreaDefaultY() | int |
| isCollision() | boolean |
| getMapId() | int[] |
| setHealth(int) | void |
| loadObjects() | void |
| setMapId(int, int) | void |

ภาพแสดงความสัมพันธ์



โครงการนี้จะมีคราสหลักทั้งหมด 15 คราสคือ

1. Class Game คือ คราสหลักที่ตัวหน้าที่สร้าง JFrame และรันเกม
2. Class GamePanel คือ คราสที่สร้าง JPanel และวาดแมพและตัวละครต่างรวมถึงควบคุมการทำงานหลักของเกมทั้งหมด
3. Class Entity คือ คราสแม่ของตัวละครต่างๆไว้ใช้สำหรับเก็บข้อมูลตำแหน่ง
4. Class Monster คือ คราสที่สืบทอดจาก Entity ไว้สำหรับประมวลผลต่างๆเกี่ยวกับ Monster
5. Class Player คือ คราสที่สืบทอดจาก Entity ไว้สำหรับประมวลผลต่างๆเกี่ยวกับผู้เล่น
6. Class PlayerState คือ คราสที่เก็บสถานะต่างๆของ Player
7. Class Objects คือ คราสที่สร้างมาเพื่อเก็บ Object ต่างๆที่เอาไว้ใส่ในเกม
8. Class AssetSetterObject คือ คราสที่สร้างมาเพื่อไว้ เซ็ตตำแหน่งของ Object ต่างๆภายในเกม
9. Class Element คือ คราสที่ไว้เก็บข้อมูลแต่ละธาตุให้กับผู้เล่น
10. Class TextureLoader คือ คราสไว้เก็บ Texture ต่างๆไว้สำหรับการวาดแมพ
11. Class KeyHandler คือ คราสใช้สำหรับดักจับ Event ต่างๆเกี่ยวกับคีย์บอร์ด
12. Class CollisionChecker คือ คราสใช้สำหรับเช็คการชนกับของ object monster หรือ map ของ player และ monster
13. Class UI คือ คราสใช้สำหรับ วาดรูปจำพวก หน้า Menu SelectMap Win Lose Pause ต่างๆ
14. Class UIStatus คือ คราสใช้สำหรับการวาดสถานะต่างๆไม่ว่าจะเป็น boxHealth หรือ boxArmor และอื่นๆ
15. Class Supermap คือ คราสใช้สำหรับเป็นคราสแม่ไว้กำหนด State ต่างๆในการวาดแมพ

2.4 รูปแบบการพัฒนาโครงการ

- ภาษา : Java
- GUI : javax.swing
- โปรแกรมวาดรูป / ตัดรูป : Photoshop, Aseprite

2.5 Constructor

```
public class Game {  
    public Game(){  
        JFrame window = new JFrame();  
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        window.setResizable(resizable:false);  
        window.setTitle(title:"Dark Knight");  
  
        GamePanel gamePanel = new GamePanel();  
        Texture.loadTexture();  
        Objects.loadObjects();  
        window.add(gamePanel);  
  
        window.pack();  
  
        window.setLocationRelativeTo(c:null);  
        window.setVisible(b:true);  
  
        gamePanel.setupGame();  
        gamePanel.startGameThread();  
    }  
    Run | Debug  
    public static void main(String[] args) {  
        new Game();  
    }  
}
```

Class Game สำหรับกำหนดส่วนประกอบต่างๆของหน้าต่าง

```
public GamePanel() {  
    this.setPreferredSize(new Dimension(screenWidth, screenHeight));  
    this.setBackground(new Color(r:16, g:16, b:16));  
    this.setDoubleBuffered(aFlag:true);  
    this.addKeyListener(keyH);  
    this.setFocusable(focusable:true);  
    this.setBackgroundGame();  
}
```

Class GamePanel ไว้สำหรับใส่ Event listener และกำหนด Background ต่างๆ

```
public KeyHandler(GamePanel gp) {  
    this.gp = gp;  
}
```

Class KeyHandler ไว้สำหรับรับ GamePanel มาเก็บไว้สำหรับดักจับเช็ค Event ต่างๆ

```
public Entity(GamePanel gp){  
    this.gp = gp;  
    imageWidth = gp.titleSize;  
    imageHeight = gp.titleSize;  
    solidArea = new Rectangle(x:0,y:0,imageWidth,imageHeight);  
}
```

Class Entity ไว้สำหรับรับ GamePanel ไว้สำหรับดักจับ Event ต่างๆและ กำหนด Default ต่างๆของ Entity ที่ทุกตัวละครต้องมี

```
public Player(GamePanel gp, KeyHandler keyH) {  
  
    super(gp);  
  
    this.keyH = keyH;  
  
    screenX = gp.screenWidth / 2 - (gp.titleSize / 2);  
    screenY = gp.screenHeight / 2 - (gp.titleSize / 2);  
  
    solidArea = new Rectangle();  
  
    setDefaultValues();  
    loadAnimation();  
    ElementEnums.loadElements();  
  
}
```

Class Player มีการรับ GamePanel และ KeyHanlder เข้ามาเพื่อดักจับ Event ต่างๆ

และมีการกำหนด ตำแหน่งของตัวละครให้อยู่ตรงกลางหน้าจอเสมอเพื่อให้ Map หรือ Object ต่างๆ Render โดนอ้างอิงจาก Player และมีกำหนด Default ค่าเริ่มต้นของ Player และมีการเรียกให้โหลด ภาพต่างๆที่ใช้สำหรับการไว้วาด Player และ Class ลูกอื่นๆที่สืบทอดมาจาก Entity ก็ทำคล้ายกันกับ Player

```
public TextureLoader(String path,boolean collision) throws IOException {  
    this.image = ImageIO.read(getClass().getResourceAsStream(path));  
    this.imageWidth = image.getWidth();  
    this.imageHeight = image.getHeight();  
    this.collision = collision;  
}
```

Class TextureLoader ไว้สำหรับรับ path และ การชน มาแล้วโหลดภาพเก็บไว้และกำหนดขนาดตัวภาพและกำหนดการชนต่างๆของ map และ Class Object ก็ทำคล้ายกันแบบนี้เพื่อเอาไว้ใช้การชนและrender ภาพต่างๆ

2.6 Encapsulation

ส่วนที่เป็น Attribute เกือบทั้งหมดจะเป็น private แต่ถ้ามีการสืบทอดบางตัวจะเป็น protected มี methods getter และ setter ในการเข้าถึง และ methods โดยส่วนใหญ่เกือบทั้งหมดจะเป็น public

2.7 Composition

```
public class CollisionChecker {
    GamePanel gp;

    public CollisionChecker(GamePanel gp) {
        this.gp = gp;
    }
}
```

คลาสต่างๆภายในเกมจะมี GamePanelเป็นส่วนประกอบภายในคลาสเกือบทุกๆคลาสเพราะต้องมีการเข้าถึงองค์ประกอบต่างๆภายในเกม เช่น

```
public Player player = new Player(this, keyH);
public ArrayList<Entity> monster = new ArrayList<Entity>();
public ArrayList<Entity> npc = new ArrayList<Entity>();
```

Class CollisionChecker ต้องมีการเข้า ถึง Player และ monster npc ต่างๆเพื่อที่จะใช้การชนของ Entity

2.8 Polymorphism

```
public ArrayList<Entity> monster = new ArrayList<Entity>();

monster.add(new FringEye(gp, x:683, y:334));

monster.add(new Mushroom(gp, x:153, y:189));
```

ถ้าเป็น Class Mushroom จะเก็บได้ Mushroom แต่ถ้าเป็น Entity สามารถเก็บ ตัวละครได้ทั้งหมดไม่ว่าจะเป็น Player หรือ Monster ต่างๆในเกมเพราะ ทุกตัวมีการสืบทอด Entity มา

2.9 Abstract

```
public abstract class Supermap implements ActionListener {

    protected GamePanel gp;
    public int MapContent[][];
    public Timer timerMap;
    public int currentTimeMap = 0;
    public int countMonster = 0;
    public int currentMonster = 0;

    public Supermap(GamePanel gp) {
        this.gp = gp;
        timerMap = new Timer(delay:1000, this);
        timerMap.start();
        setDefaultSpawnEntityAndObjects();
        setDefaultObjects();
        resetCountKilled();
    }

    abstract public void update();

    abstract public void setDefaultObjects();
}
```

Class Supermap เป็น abstract class ใช้บอกว่าต้องมี methods อะไรบ้างในแต่ละหน้าที่จำเป็น เช่น คราสลูกต้องมี method update กับ setDefaultObjects เพื่อกำหนด monster และ object ที่จะต้อง spawn ในแต่ละ map เพราะในแต่ละ map มี การวาด Object และ Spawn Monster ที่ต่างกัน

2.10 Inheritance

```
public abstract class Supermap implements ActionListener {
```

```
public class M1_ST1 extends Supermap {
```

```
public class M1_ST2 extends Supermap{
```

Class ต่างๆที่ทำงานเกี่ยวกับการ Map จะถูกสืบทอดมาจาก Supermap เพราะแต่ละ Map ต้องมี Texture ในการวาด map แต่ข้างในการเปลี่ยนและมี event ในการทำงานที่แตกต่างกัน

```
public abstract class Entity {}
```

```
public class Player extends Entity {
```

```
public abstract class Monster extends Entity implements Runnable{
```

```
public class Mushroom extends Monster{
```

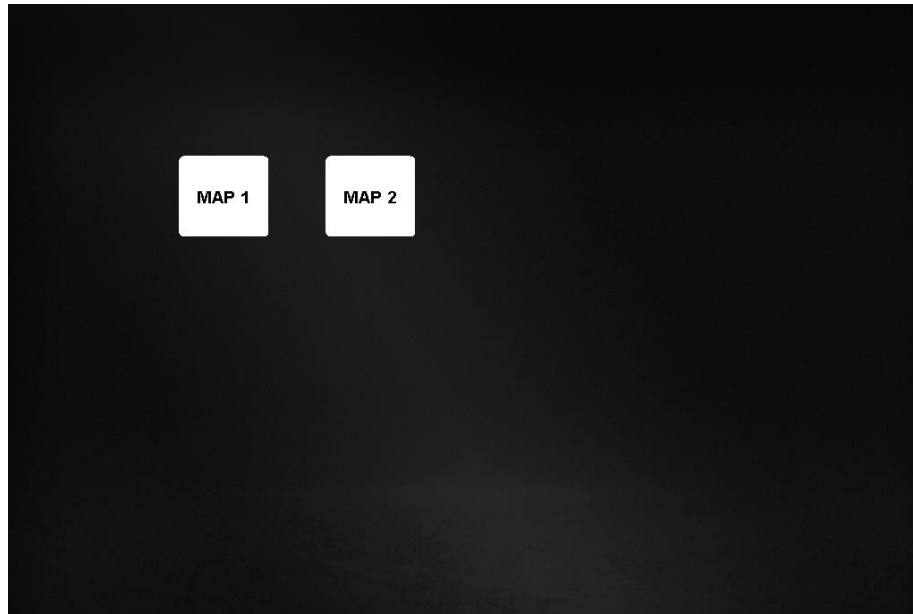
Class ที่มีการทำงานเกี่ยวกับตัวละครจะถูกสืบทอดมาจาก Class Entity และ สำหรับ Monster บางตัว อาจถูกสืบทอดโดย Monster เพราะ Monster บางตัวมีความสามารถพิเศษมากกว่า Monster ทั่วไป และ Class Player สืบทอดจาก Entity เพราะ Player มีความสามารถพิเศษมากกว่า Entity ทั่วไป และ ต่างจากพวก Monster

2.11 GUI



หน้าเริ่มเกม

ประกอบไปด้วยปุ่ม START สำหรับเริ่มเกม และ ปุ่ม EXIT สำหรับปิดโปรแกรม และ background สำหรับภาพพื้นหลัง



หน้าเลือกด่าน

ประกอบไปด้วยปุ่มสำหรับเลือกเล่นในแต่ละด่าน และมี background สำหรับภาพพื้นหลัง



หน้าเล่นเกม

ประกอบไปด้วย Texture ของ map และ Object หลอดเลือด หลอดเกราะ ชาติ mana และ
ส่วนประกอบของฉากตัวละครแต่ละตัว

2.12 Event handling

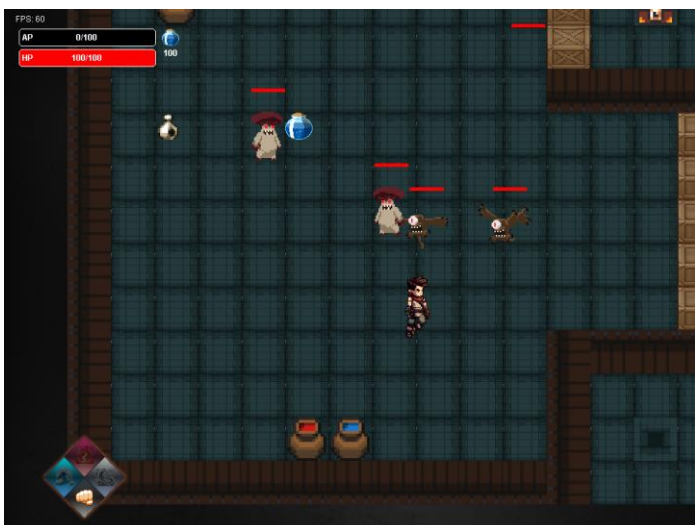


```
// MouseListener
@Override
public void mouseClicked(MouseEvent e) {
    if (gp.gameState == gp.menuState) {
        if (e.getX() > gp.screenWidth / 2 - 100 && e.getX() < gp.screenWidth / 2 + 100 && gp.titleSize*3 {
            if (e.getY() > gp.screenHeight - gp.titleSize*3 && e.getY() < gp.screenHeight - gp.titleSize*2) {
                gp.gameState = gp.selectMapState;
            } else if (e.getY() > gp.screenHeight - gp.titleSize*2 && e.getY() < gp.screenHeight - gp.titleSize) {
                System.exit(status:0);
            }
        }
    } else if (gp.gameState == gp.selectMapState) {
        if (e.getX() > gp.titleSize*3 && e.getX() < gp.titleSize*3 + 100) {
            if (e.getY() > gp.titleSize*3 && e.getY() < gp.titleSize*3 + 100) {
                gp.player.setMap(parent:0, child:0);
                gp.map = new M1_S11(gp);
                gp.gameState = gp.playerState;
            }
        } else if (e.getX() > gp.titleSize*3 + 100 && e.getX() < gp.titleSize*4 + 200) {
            if (e.getY() > gp.titleSize*3 && e.getY() < gp.titleSize*3 + 100) {
                gp.player.setMap(parent:1, child:0);
                gp.map = new M2_S11(gp);
                gp.gameState = gp.playerState;
            }
        }
    }
}

} else if (gp.gameState == gp.gameOverState || gp.gameState == gp.gameWinState || gp.gameState == gp.pauseState) {
    if (e.getX() > gp.screenWidth / 2 - gp.titleSize*2 && e.getX() < gp.screenWidth / 2 + gp.titleSize*2) {
        if (e.getY() > gp.titleSize*0 && e.getY() < gp.titleSize*0 + gp.titleSize) {
            gp.resetStateAll();
            if (gp.player.getStateMap()[0] == 0) {
                gp.map = new M1_S11(gp);
            } else if (gp.player.getStateMap()[0] == 1) {
                gp.map = new M2_S11(gp);
            }
            gp.gameState = gp.playerState;
        } else if (e.getY() > gp.titleSize*7 + 15 && e.getY() < gp.titleSize*7 + 15 + gp.titleSize) {
            gp.gameState = gp.menuState;
            gp.resetStateAll();
        }
    }
}
}
```

```
gp.addMouseListener(this);
```

ใช้ MouseListener ในการหาตำแหน่งของเมาส์และดักจับการคลิกเมาส์เพื่อเอาตำแหน่งไปเทียบกับว่าอยู่ตำแหน่งที่เราสนใจหรือไม่ มีการใช้อยู่หน้าเมนู หน้าเลือกด่าน หน้าชนะ หน้าแพ้ หน้าpause ต่างๆ



```
@Override
public void keyReleased(KeyEvent e) {
    int code = e.getKeyCode();

    if (code == KeyEvent.VK_W) {
        up = false;
    }
    if (code == KeyEvent.VK_A) {
        left = false;
    }
    if (code == KeyEvent.VK_S) {
        down = false;
    }
    if (code == KeyEvent.VK_D) {
        right = false;
    }
}
```

```
this.addKeyListener(keyH);
```


ในหน้าเล่นเกมจะมีการรับการกดปุ่มต่างๆเพื่อใช้ควบคุมตัวละครและบังคับไปทิศทางต่างๆ โดยใช้ KeyListener ในการดักจับ

2.13 อัลกอริทึมที่สำคัญในโปรแกรม

```
public void checkMap(Entity entity) {
    try {
        int entityLeftWorldX = entity.worldX + entity.solidArea.x;
        int entityRightWorldX = entity.worldX + entity.solidArea.x + entity.solidArea.width;
        int entityTopWorldY = entity.worldY + entity.solidArea.y;
        int entityBottomWorldY = entity.worldY + entity.solidArea.y + entity.solidArea.height;

        int entityLeftCol = entityLeftWorldX / gp.tileSize;
        int entityRightCol = entityRightWorldX / gp.tileSize;
        int entityTopRow = entityTopWorldY / gp.tileSize;
        int entityBottomRow = entityBottomWorldY / gp.tileSize;

        Texture titleNum1, titleNum2;
        if (entity.direction == "up") {
            entityTopRow = (entityTopWorldY - entity.speed) / gp.tileSize;
            titleNum1 = Texture.values()[gp.map.MapContent[entityTopRow][entityLeftCol]];
            titleNum2 = Texture.values()[gp.map.MapContent[entityTopRow][entityRightCol]];
            if (titleNum1.isCollision() || titleNum2.isCollision()) {
                entity.collisionOn = true;
            }
        }
        if (entity.direction == "down") {
            entityBottomRow = (entityBottomWorldY + entity.speed) / gp.tileSize;
            titleNum1 = Texture.values()[gp.map.MapContent[entityBottomRow][entityLeftCol]];
            titleNum2 = Texture.values()[gp.map.MapContent[entityBottomRow][entityRightCol]];
            if (titleNum1.isCollision() || titleNum2.isCollision()) {
                entity.collisionOn = true;
            }
        }
        if (entity.direction == "left") {
            entityLeftCol = (entityLeftWorldX - entity.speed) / gp.tileSize;
            titleNum1 = Texture.values()[gp.map.MapContent[entityTopRow][entityLeftCol]];
            titleNum2 = Texture.values()[gp.map.MapContent[entityBottomRow][entityLeftCol]];
            if (titleNum1.isCollision() || titleNum2.isCollision()) {
                entity.collisionOn = true;
            }
        }
        if (entity.direction == "right") {
            entityRightCol = (entityRightWorldX + entity.speed) / gp.tileSize;
            titleNum1 = Texture.values()[gp.map.MapContent[entityTopRow][entityRightCol]];
            titleNum2 = Texture.values()[gp.map.MapContent[entityBottomRow][entityRightCol]];
            if (titleNum1.isCollision() || titleNum2.isCollision()) {
                entity.collisionOn = true;
            }
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        entity.collisionOn = true;
    }
}
```

ใช้ในการตรวจสอบการชนกันระหว่างตัวละครกับฉากภายในเกม

```

public int checkEntity(Entity entity, ArrayList<Entity> target) {
    int index = -1;

    for (int i = 0; i < target.size(); i++) {
        if (target.get(i) != null) {
            // Get entity's solid area position
            entity.solidArea.x = entity.worldX + entity.solidArea.x;
            entity.solidArea.y = entity.worldY + entity.solidArea.y;
            // Get the target entity's solid area position
            target.get(i).solidArea.x = target.get(i).worldX + target.get(i).solidArea.x;
            target.get(i).solidArea.y = target.get(i).worldY + target.get(i).solidArea.y;

            switch (entity.direction) {
                case "up":
                    entity.solidArea.y = entity.solidArea.y - entity.speed;
                    break;
                case "down":
                    entity.solidArea.y = entity.solidArea.y + entity.speed;
                    break;
                case "left":
                    entity.solidArea.x = entity.solidArea.x - entity.speed;
                    break;
                case "right":
                    entity.solidArea.x = entity.solidArea.x + entity.speed;
                    break;
                default:
                    break;
            }

            if (entity.solidArea.intersects(target.get(i).solidArea)) {
                if (target.get(i) != entity) {
                    entity.collisionOn = true;
                    index = i;
                }
            }

            entity.solidArea.x = entity.solidAreaDefaultX;
            entity.solidArea.y = entity.solidAreaDefaultY;
            target.get(i).solidArea.x = target.get(i).solidAreaDefaultX;
            target.get(i).solidArea.y = target.get(i).solidAreaDefaultY;
        }
    }

    return index;
}

```

ใช้ในการตรวจสอบว่าตัวละครชนกันหรือไม่

```

public ArrayList<Integer> checkPlayerAttackMonster(Entity entity, ArrayList<Entity> target) {
    ArrayList<Integer> monsterHit = new ArrayList<>();

    for (int i = 0; i < target.size(); i++) {
        Entity monster = target.get(i);
        if (monster != null) {
            // เก็บค่าตำแหน่งเดิมไว้
            int originalAttackX = entity.attackArea.x;
            int originalAttackY = entity.attackArea.y;
            int originalSolidX = monster.solidArea.x;
            int originalSolidY = monster.solidArea.y;

            // คำนวณตำแหน่งใหม่
            entity.attackArea.x += entity.worldX;
            entity.attackArea.y += entity.worldY;
            monster.solidArea.x += monster.worldX;
            monster.solidArea.y += monster.worldY;

            // ตรวจสอบการชนกัน
            if (entity.attackArea.intersects(monster.solidArea)) {
                if (monster != entity) {
                    monsterHit.add(i);
                }
            }

            // คืนค่าตำแหน่งเดิม
            entity.attackArea.x = originalAttackX;
            entity.attackArea.y = originalAttackY;
            monster.solidArea.x = originalSolidX;
            monster.solidArea.y = originalSolidY;
        }
    }

    return monsterHit;
}

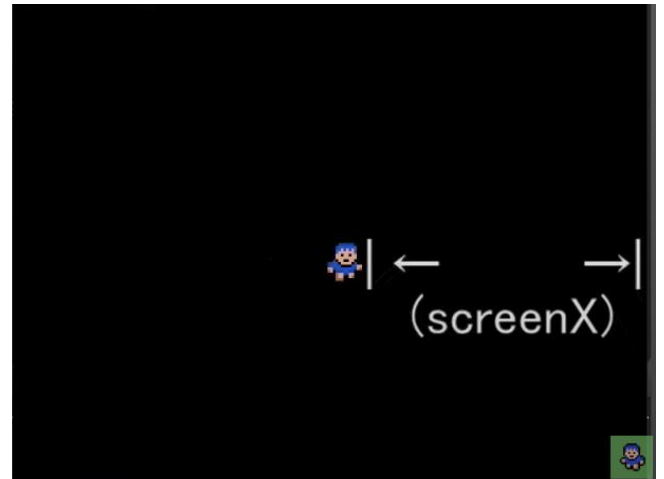
```

ใช้ในการตรวจสอบว่าตอนผู้เล่นโจมตีมี Entity ไหนที่โดนดาบหรือไม่

```

MapContentet = new int[][]{
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,15,15,15,15,15,15,15,15,15,15,15,15,15,0},
    {0,15,7,7,7,7,15,15,15,15,15,7,7,7,15,0},
    {0,15,7,7,7,7,15,7,7,7,7,15,15,7,7,15,0},
    {0,15,7,7,7,15,7,7,7,7,7,15,15,7,7,15,0},
    {0,15,7,7,15,7,7,7,7,7,7,15,7,7,15,0},
    {0,15,7,7,7,7,7,7,7,7,7,7,7,7,15,0},
    {0,15,7,7,7,7,7,7,7,7,7,7,7,7,15,0},
    {0,15,7,7,15,7,7,7,7,7,7,7,7,7,7,15,0},
    {0,15,7,7,7,15,7,7,7,7,7,15,15,7,7,15,0},
    {0,15,7,7,7,15,7,7,7,7,7,15,15,7,7,15,0},
    {0,15,7,7,7,15,7,7,7,7,7,15,15,7,7,15,0},
    {0,15,7,7,7,15,15,15,15,15,15,7,7,7,15,0},
    {0,15,15,15,15,15,15,15,15,15,15,15,15,15,0}
};

```



```

public void draw(Graphics g2) {
    for (int i = 0; i < MapContentet.length; i++) {
        for (int j = 0; j < MapContentet[i].length; j++) {
            int worldX = j * gp.titleSize;
            int worldY = i * gp.titleSize;
            int screenX = worldX - gp.player.worldX + gp.player.screenX;
            int screenY = worldY - gp.player.worldY + gp.player.screenY;

            if (worldX + gp.titleSize > gp.player.worldX - gp.player.screenX &&
                worldX - gp.titleSize < gp.player.worldX + gp.player.screenX &&
                worldY + gp.titleSize > gp.player.worldY - gp.player.screenY &&
                worldY - gp.titleSize < gp.player.worldY + gp.player.screenY) {
                Texture texture = Texture.values()[MapContentet[i][j]];
                g2.drawImage(texture.getTexture(), screenX, screenY, gp.titleSize, gp.titleSize, observer:null);
            }
        }
    }
}

```

ใช้ในการ วาดmap จาก MapContent ที่เป็น Array2D ที่มีจากเก็บ id ของ Texture เอาไว้ และในการวาดแมพให้อ้างอิงจาก Player ให้อยู่ตรงกลางตลอดเวลาและให้ map render เฉพาะ สโคปที่ผู้เล่นมองเห็นลดการกินทรัพยากร

บทที่ 3

สรุป

3.1 ปัญหาที่พบระหว่างการพัฒนา

เนื่องจากเกมมีองค์ประกอบหลายอย่างทำให้การพัฒนาในช่วงแรกอาจเกิดความสับสนใจและไม่เข้าใจในบางส่วน และ ระหว่างการศึกษาข้อมูล หลังจากศึกษาไประยะเวลาหนึ่งทำให้สามารถเข้าใจได้ง่ายมากขึ้น

3.2 จุดเด่นของโปรแกรม

ผู้เล่นจะได้สวมบทเป็นอัศวินแห่งความมืดที่ได้รับดาบจากพระเจ้าไปกำจัดปีศาจในระหว่างทางผู้เล่นต้องเอาตัวรอดจาก monster ในแต่ละด่านซึ่งในแต่ละด่านและแต่ละ monster มีจุดเด่นของตัวเอง โดนผู้เล่นต้องใช้ความสามารถของดาบแต่ละธาตุที่มีมาให้ในการกำจัด monster นั้นๆจนไปถึงboss ถ้าผู้เล่นชนะ boss และกำจัด monster ทั้งหมดได้ถึงจะชนะ

3.3 คำแนะนำสำหรับผู้สอน

อยากให้อาจารย์มาเฉลยแลปต่างๆ เพราะในแต่ละแลปส่งไปผลลัพธ์ตรงแต่ไม่แน่ใจว่าถูกตาม logic ของมันจริงๆไหม หรือ ไม่ก็มี grader เอาไว้ตรวจ เนื่องจากบางข้ออัลกอริทึมที่ยากๆไม่มั่นใจว่าจะถูกจริงๆหรือเปล่าครับ