

Tower Defense Documentation

Create by

Chotipat Assawapayukul 6330116621

Thanadol Chitthamlard 6330216721

2110215 Programming Methodology

Semester 1 Academic year 2021

Department of Computer Engineering

Faculty of Engineering, Chulalongkorn University

Tower Defense

Introduction

Tower Defense เป็นเกมแนว endless ที่ว่าด้วยเรื่องของการวางป้อมปราการเพื่อปกป้องป้อมของแม่ทัพ (ผู้เล่น) เพื่อไม่ให้ Monster เข้ามาโจมตีป้อมแม่ทัพได้ ดังนั้นด้วยแนวคิดของเกมนี้รวมกับการที่เคยเล่นเกมนี้มาก่อนทำให้พวกผมได้รับแรงบันดาลใจในการสร้างเกมมาจาก Bloon battle TD (BTD 5) และ Tower Defense

Tower Character

ในเกมนี้เราจะมี 3 ป้อมพื้นฐาน ก่อนทำการ upgrade ป้อมไป level สูงสุดแล้วนั้น ผู้เล่นต้องทำการเลือกความสามารถของป้อม (โดยเลือกความสามารถ 1 จาก 2 ของความสามารถของป้อม) ซึ่งในเกมนี้เราจะแบ่งป้อมได้ทั้งหมด 9 ประเภท

- Base Tower Type1 ป้อมนี้จะเป็น 1 ใน 3 ของป้อมเริ่มต้นของเกมซึ่งเมื่อ upgrade ป้อมนี้ไปจนถึง level สูงสุดแล้ว ผู้เล่นต้องเลือกว่าจะให้ป้อมนี้ upgrade ไปเป็น Fire หรือ Ice ซึ่งสองป้อมนี้นั้นก็จะมีความสามารถแตกต่างกัน
- Base Tower Type2 ป้อมนี้จะเป็น 1 ใน 3 ซึ่งถ้า upgrade ไปจนถึง level สูงสุดนั้นผู้เล่นก็ต้องเลือกอีกเช่นกันว่าจะให้เป็น Farm หรือ Strength ซึ่งสองป้อมนี้จะไม่สามารถโจมตี monster ได้ (แต่ก่อนที่จะ upgrade นั้นสามารถโจมตี monster ได้ปกติ) แต่จะมีความสามารถที่แตกต่างอีกเช่นกัน
- Base Tower Type3 ป้อมนี้จะเป็น 1 ใน 3 ซึ่งถ้า upgrade ไปจนถึง level สูงสุดนั้นผู้เล่นก็ต้องเลือกอีกเช่นกันว่าจะให้เป็น Boom หรือ Laser ได้แต่จะมีความสามารถที่แตกต่างอีกเช่นกัน
- Fire เป็นป้อมที่ upgrade มาจาก Base Tower Type1 ซึ่งความสามารถของป้อมนี้จะโจมตี monster แรงกว่าป้อมปกติ
- Ice เป็นป้อมที่ upgrade มาจาก Base Tower Type1 ซึ่งความสามารถของป้อมนี้จะทำให้ monster ที่ถูกป้อมนี้โจมตีนั้นเคลื่อนที่ช้าลง
- Farm เป็นป้อมที่ upgrade มาจาก Base Tower Type2 แม้ว่าป้อมนี้นั้นไม่สามารถโจมตี monster ได้ แต่ป้อมนี้จะทำการผลิตเงินเพื่อให้กับผู้เล่นได้
- Strength เป็นป้อมที่ upgrade มาจาก Base Tower Type2 จะคล้ายกับ Farm ที่โจมตี monster ไม่ได้ แต่จะทำให้ Tower ต่าง ๆ ที่อยู่ขอบเขตของป้อมนี้มีพลังการโจมตีที่เพิ่มขึ้น
- Laser เป็นป้อมที่ upgrade มาจาก Base Tower Type3 จะมีการโจมตีแบบปกติแต่กระสุนที่ยิงออกไปนั้นจะทะลุ monster ได้

- Boom เป็นป้อมที่ upgrade มาจาก Base Tower Type3 จะมีการโจมตีเป็นวงกว้าง ซึ่ง monster ทุกตัวที่อยู่ในขอบเขตของป้อมนี้จะได้รับ Damage จากการโจมตีของป้อมนี้หมดทุกตัว



รูป Base Tower Type1 , Fire และ Ice ตามลำดับ



รูป Base Tower Type2 , Farm และ Strength ตามลำดับ



รูป Base Tower Type3 , Boom และ Laser ตามลำดับ

Monster Character

ในเกมนี้จะแบ่ง monster ได้หลัก ๆ ได้ 2 ประเภท คือ ลูกสมุน กับ หัวหน้า (boss) ซึ่งลูกสมุนนั้นก็แบ่งได้อีก 2 ประเภท คือ วิ่งช้า (soldier) กับ วิ่งเร็ว (speedsoldier) แต่ของหัวหน้านั้นจะมีแค่วิ่งช้า ซึ่งความแตกต่างระหว่างสองประเภทนี้คือหัวหน้านั้นจะมีเลือดที่มากกว่าลูกสมุน และ ถ้าหัวหน้าเข้าไปในป้อมแม่ทัพเราได้เลือดของป้อมแม่ทัพนั้นจะลดเยอะกว่าลูกสมุน

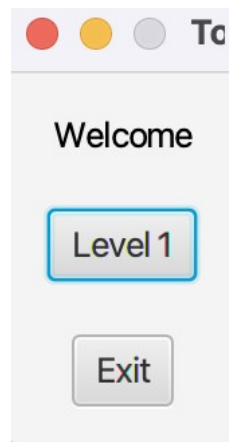


รูป soldier , speedsoldier และ boss ตามลำดับ

Gameplay

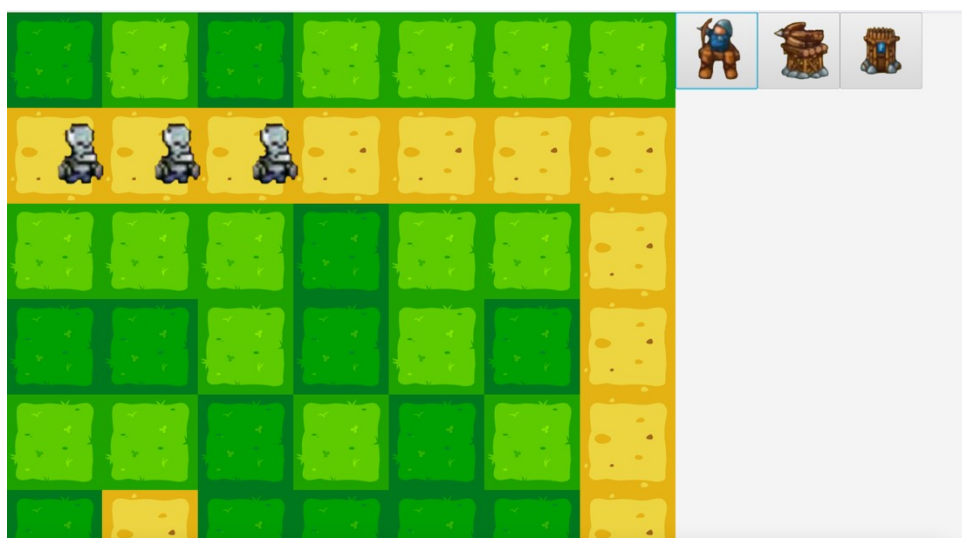
สิ่งที่คุณผู้เล่นจะมีตั้งแต่ต้นเกมคือ เงินที่มีค่าเริ่มต้น 500 (ในเกมนี้เงินจะได้มากจากการที่ป้อม Farm ผลิตเงิน หรือ ตามเวลาที่กำหนด หรือ การขายป้อมของเรา) และ เลือดของป้อมแม่ทัพ (เลือดของผู้เล่น) จะมีค่าเริ่มต้น คือ 150 ซึ่งค่านี้จะลดลงได้อย่างเดียว ซึ่งระบบของเกมนี้จะมาเป็นรอบ (round) การที่ round เพิ่มขึ้นนั้นก็ส่งผลกระทบต่อจำนวน monster หรือ ประเภทของ monster ที่จะมาใน round นั้น ๆ ซึ่งจุดหมายของเกมนี้คือการที่ป้อมทั้งหมดในเกมต้องพยายามจัดการ monster ไม่ให้เข้ามาในป้อมแม่ทัพได้ ถ้า monster เข้าไปจะทำการลดเลือดของผู้เล่น และเมื่อเลือดของผู้เล่นเป็น 0 หรือน้อยกว่าจะจบเกมทันที

Example



รูปเมื่อ *Run* เกม
ขึ้นมา

ตอนเริ่มเกมขึ้นนั้นจะพบว่ามี monster จะเข้ามาทางด้านซ้ายตามทางเดินสี่เหลี่ยมของเกมส่วนทางด้านขวานั้นจะมี Base Tower Type ทั้ง 3 ตัวเริ่มต้นก่อน โดยใช้เมาส์คลิกเพื่อเลือก Tower และคลิกอีกทีเพื่อวางตัวใน Map แต่จะวางขวางทางเดิน (ทางเดินสี่เหลี่ยม) ของ monster ไม่ได้



เมื่อผู้เล่นใช้เมาส์คลิก Tower บน Map แล้วจะเห็นว่ามียวงกลมสีเทาล้อมรอบส่วนนั้นคือขอบเขตของ Tower ที่จะเห็น monster ใน Map และข้อมูลขึ้นที่ฝั่งขวามืออยู่คือ properties ของ Tower นั้นจะบอก Attack , SpeedAttack , level , Price , Range และ BulletsType จะพบว่าส่วนข้างล่างของ properties จะมีปุ่มสามปุ่ม คือ upgrade ฝั่งซ้าย , upgrade ฝั่งขวา และ sell (คือการขาย Tower นั้นทิ้ง) การ upgrade นั้นจำเป็นต้องใช้เงินตามที่ปุ่มนั้นระบุไว้ (ถ้าไม่ได้จะไม่สามารถ upgrade ได้) ส่วนด้านล่างของปุ่ม คือ จำนวนรอบ (round) ที่เริ่มไปได้ ซึ่งจะเพิ่มขึ้นไปเรื่อย ๆ ส่วนต่อมาคือ lifePoint จะเป็นเลือดของผู้เล่นเองซึ่งจะเริ่มที่ 150 และเมื่อถึงน้อยกว่าเท่ากับ 0 เมื่อไหร่ก็จะจบเกมทันที ส่วนสุดท้ายคือ Current Money จะเป็นเงินของผู้เล่นใช้สำหรับ การซื้อ Tower ใหม่ละ Upgrade Tower

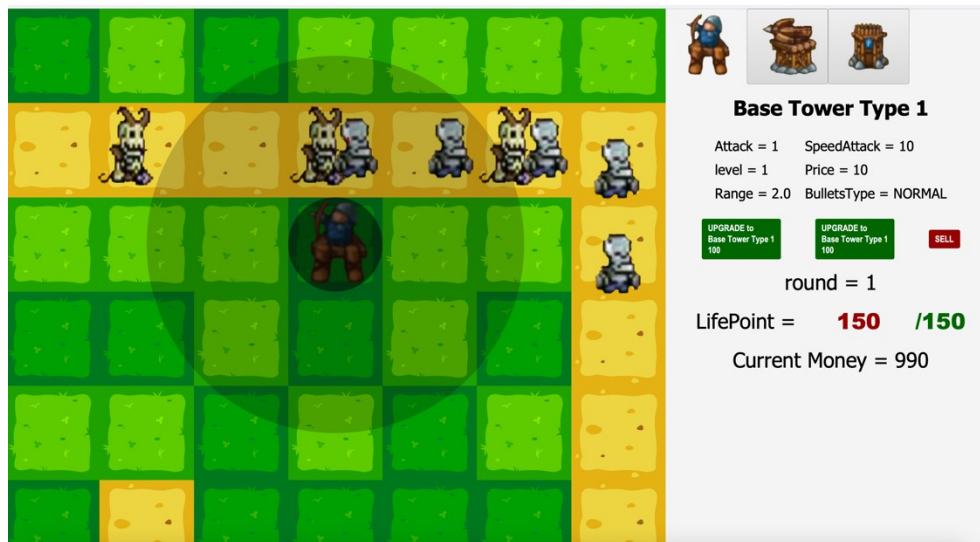
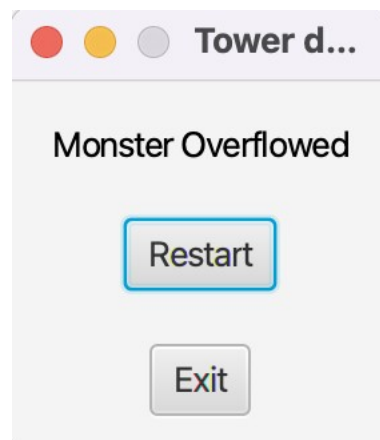
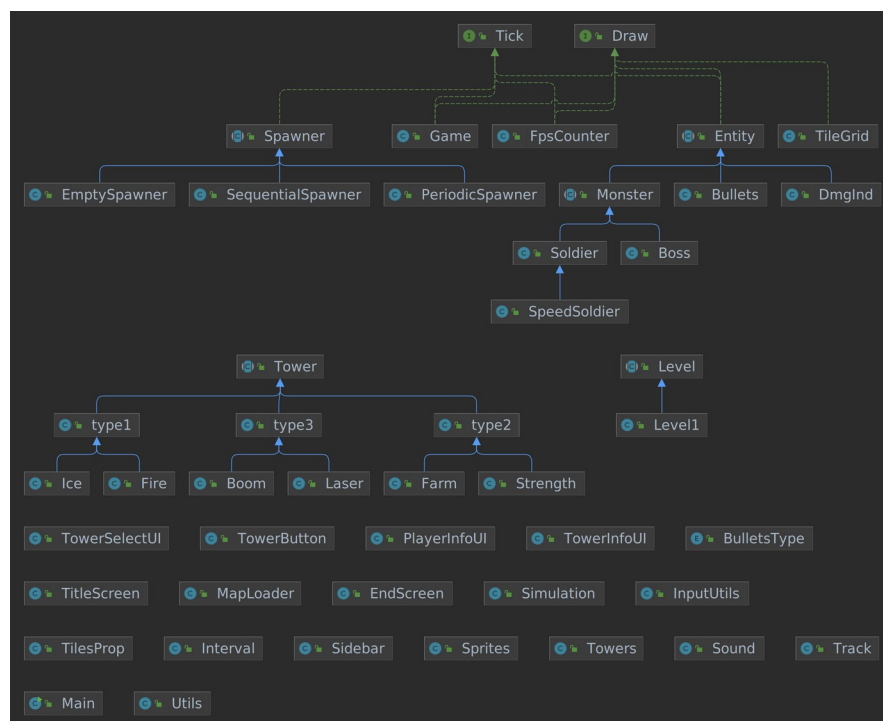
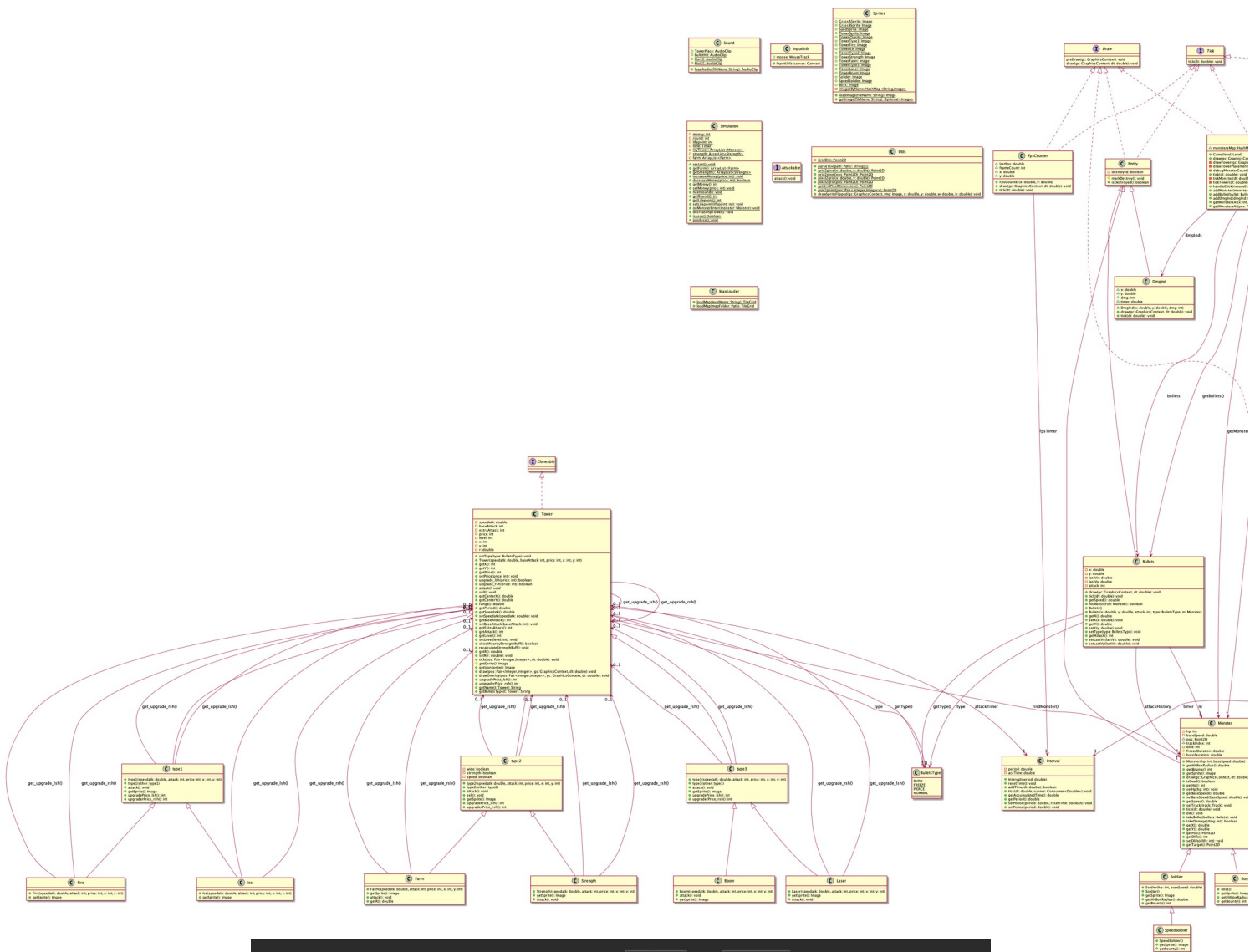


Figure 2: รูปตอนคลิกเมาส์ไปที่ Tower ที่ผู้เล่นเลือก



รูปตอนจบเกม

UML Diagram



1.Package entity.base

1.1 Class Bullets extends Entity

Class นี้แสดงถึงกระสุนในเกมที่จะออกมาตอน Tower ยิง Monster ในเกม

1.1.1 Field

- int attack	ค่าพลังการโจมตีของลูกกระสุนซึ่งมาจาก attack ของ Tower ที่ยิงกระสุน
- Monster m	เก็บค่า Monster ที่เป็นเป้าหมายของลูกกระสุน ส่วนมากจะเป็น Monster ที่อยู่ใกล้ Tower มากที่สุดในจังหวะที่ยิง
- BulletsType type	เก็บค่า BulletsType ของกระสุน
- int x	เก็บค่าตำแหน่ง x ของ Bullets นั้นในเกม
- int y	เก็บค่าตำแหน่ง y ของ Bullets นั้นในเกม
- double lastVx	เก็บความเร็วล่าสุดในแกน x ใช้กำหนดทิศการเคลื่อนที่ในกระสุนแบบ BulletsType.PIERCE
- double lastVy	เก็บความเร็วล่าสุดในแกน y ใช้กำหนดทิศการเคลื่อนที่ในกระสุนแบบ BulletsType.PIERCE
- double HashSet<Monster> attackHistory	ใช้ในกระสุนแบบ BulletsType.PIERCE ไว้เก็บ Monster ที่เคยถูกกระสุนโจมตีได้แล้วเพื่อป้องกันไม่ให้โจมตีตัวเดิมซ้ำหลายรอบ

1.1.2 Constructor

+ Bullets (int x , int y , int attack, BulletsType type, Monster m)	สร้าง Bullets จาก field ทั้งหมดที่ให้มา
--	---

1.1.3 Method

+ void draw (Graphics gc , double dt)	ทำการวาดวงกลม และปรับสีแสดงถึงชนิดของกระสุน สีดำเป็น BulletsType.NORMAL สีฟ้าเป็น BulletsType.FREEZE สีแดงเป็น BulletsType.BURN และสีม่วงเป็น BulletsType.PIERCE
+ void tick (double dt)	ทำให้กระสุนเคลื่อนที่ไปหา Monster m โดยที่ถ้ากระสุนไปใกล้ Monster m แล้วจะหายไป แต่ถ้ากระสุนเป็นแบบ PIERCE จะเคลื่อนที่เป็นเส้นตรง ทะลุและทำดาเมจใส่ Monster และจะหายไปตอนออกจาก Map

+ Boolean hitMonster (Monster m)	จะตรวจว่าถ้ากระสุนเข้าใกล้ Monster m ถ้ากระสุนห่างจาก Monster m น้อยกว่า 0.5 จะทำการคือค่า true ถ้าไม่คือค่า false
+ setters and getter each field as needed	

1.2 enum BulletsType

เป็นประเภทของกระสุนทั้งหมด จะมีอยู่ 4 ประเภท คือ BURN , FREEZE , NORMAL และ

PIERCE

1.3 Abstract Class Entity implements Draw, Tick

1.3.1 Field

- Boolean destroyed	เก็บว่า Entity ควรถูกทำลายไหม
---------------------	-------------------------------

1.3.2 Method

# void markDestroy ()	ตั้งค่าให้ destroyed เป็น true
+ Boolean isDestroyed ()	Return destroyed

1.4 Abstract Class Monster extends Entity

Class นี้เป็น class แม่ของ Monster ทั้งหมดในเกมนี้

1.4.1 Field

- int hp	เก็บเลือดของ Monster
- int speed	เก็บค่าความเร็วในการเดินของ Monster
- Track track	เส้นทางที่ Monster เคลื่อนที่
Point2D pos	ตำแหน่งกึ่งกลางของ Monster
- int trackIndex	เก็บว่า Monster เคลื่อนที่ไปเท่าไรแล้วใน track
- int dlife	เก็บค่าที่จะไปลด lifepoint ของผู้เล่น กำหนดให้เป็น 1
# double freezeDuration	ระยะเวลาที่จะโดนผลของ BulletsType.FREEZE
# double burnDuration	ระยะเวลาที่จะโดนผลของ BulletsType.BURN

1.5.2 Constructor

+ Monster (int hp , int speed)	สร้าง Monster จาก hp และ speed
----------------------------------	--------------------------------

1.5.3 Method

+ Boolean isDead ()	คืนค่า true ถ้า hp ของ Monster มีค่าน้อยกว่าเท่ากับ 0 และคือค่า false ในอีกกรณี
+ void draw (GraphicsContext gc , double dt)	วาดภาพของ Monster ลงไปในเกมตามตำแหน่ง pos ภาพเอามาจาก getSprite() ซึ่งจะเป็นภาพหันหน้าไปทางขวา หาก Monster เคลื่อนที่ไปทางซ้ายจะทำการกลับภาพตามแกน x ก่อนวาด
+ void tick (double dt)	1. ทำให้ Monster เคลื่อนที่ไปตามเส้นทาง Track 2. ลด burnDuration กับ freezeDuration ตามเวลา
+ void die ()	จะเรียก method markDestroy () และเพิ่มเงินผู้เล่นเป็นจำนวนเท่ากับ getBounty()
+ boolean takeDamage(int dmg)	ทำให้ monster นี้โดน damage คือเสียเลือด. หากเลือดเหลือน้อยกว่าเท่ากับ 0 มันจะตาย return boolean, แสดงว่าหลังโดน damage ตายไหม
+ boolean takeBullet(Bullets bullets)	ประมวลผลเมื่อ monster โดน bullet ยิงโดน. จัดการเรื่องโดน damage และสถานะ freeze/burn จาก bullet.
+ setters and getter of each field as needed	

1.6 Abstract Class Tower implement Cloneable

Class นี้แสดงถึง Tower ทั้งหมดของเกมนี้

1.6.1 Field

- double speedatk	เก็บค่าความเร็วในการโจมตี Monster ของ Tower
- int baseAttack	เก็บค่าพลังในการโจมตี Monster ของ Tower
- int extraAttack	เก็บค่าพลังในการโจมตี Monster ที่มาจากผลอยู่ในระยะของ Strength Tower
- int price	เก็บราคาของ Tower
- int level	เก็บค่า level ของ Tower กำหนดให้ตอนแรกเท่ากับ 1
- int x	เก็บค่าตำแหน่ง x ของ Tower นั้นในเกม

- int y	เก็บค่าตำแหน่ง y ของ Tower นั้นในเกม
- int r	เก็บรัศมีในการโจมตีของ Tower
- BulletsType type	เก็บประเภทของกระสุนของ Tower
- Interval attackTimer	Timer ควบคุมช่วงการโจมตีของ Tower

1.6.2 Constructor

+ Tower(int speedatk , int attack , int price , int x , int y)	สร้าง Tower จาก speedatk , attack , price , x และ y
--	---

1.6.3 Method

+ Optional<Tower> get_upgrade_lsh ()	การ upgrade ฝั่งซ้ายของ Tower
+ Optional<Tower> get_upgrade_rsh ()	การ upgrade ฝั่งขวาของ Tower
+ void attack ()	โจมตี Monster
+ int upgradePrice_lsh ()	การเพิ่มราคาในการ upgrade ฝั่งซ้ายของ Tower
+ int upgradePrice_rsh ()	การเพิ่มราคาในการ upgrade ฝั่งขวาของ Tower
+ void tick (Pair<Integer, Integer> pos , double dt)	เรียกใช้ method attack () ด้วยความถี่ซึ่งคือ speedatk
+ void draw (Pair<Integer,Integer>)	วาดรูปของ Tower
+ void drawOverlay (Pair<Integer,Integer> , GraphicsContext gc , double dt)	แสดงขอบเขตของ Tower นั้น ๆ
+ Monster findMonster ()	ใช้หา Monster ที่อยู่ใกล้ Tower มากที่สุด และ Monster ตัวนั้นต้องอยู่ในขอบเขตของ Tower ถ้าพบให้คืนค่า Monster ตัวนั้นมา แต่ถ้าไม่พบให้คืนค่า null แทน
+ String getName (Tower t)	ตรวจสอบว่า Tower t นั้นอยู่ใน class ไหน ถ้าอยู่ใน class Fire ให้คืนค่า “Fire” class Ice ให้คืนค่า “Ice”

	Class Boom คื้นค่า “Boom” class Laser คื้นค่า “Laser” Class Farm คื้นค่า “Farm” class Strength คื้นค่า “Strength” Class type1 คื้นค่า “Base Tower Type 1” Class type2 คื้นค่า “Base Tower Type 2” และ class type3 คื้นค่า “Base Tower Type3”
+ inRangeType2 ()	ทำการตรวจว่า Tower ไหนบ้างที่อยู่ในขอบเขตของ Strength ถ้าอยู่ให้เพิ่มพลังในการโจมตีไปอีก 100 แต่ถ้าไม่ไม่ต้องทำอะไร
+ double range ()	ทำการคำนวณโดยหาพื้นที่จาก r โดยพื้นที่นั้นเป็นวงกลม
+ void sell ()	คือการลบ Tower นั้นและเราจะได้เงินคื้นเท่ากับราคาของ Tower นั้นหาร 10
+ boolean upgrade_lsh (int price)	ทำการ upgrade Tower ฝั่งซ้ายโดย ถ้าเงินของผู้เล่นน้อยกว่า price ให้คื้นค่า false แต่ถ้ามากกว่าให้ลดเงินของผู้เล่นเท่ากับ price และสร้าง Tower ที่เรา upgrade ขึ้นมาแทน
+ boolean upgrade_rsh (int price)	ทำการ upgrade Tower ฝั่งขวาโดย ถ้าเงินของผู้เล่นน้อยกว่า price ให้คื้นค่า false แต่ถ้ามากกว่าให้ลดเงินของผู้เล่นเท่ากับ price และสร้าง Tower ที่เรา upgrade ขึ้นมาแทน
+ boolean checkNearbyStrengthBuff()	คำนวณว่า tower อยู่ในวง buff จาก tower แบบ strength ใหม่
+ void recalculateStrengthBuff()	คำนวณ extraDamage ที่ได้จาก strength tower ใหม่.
+ setters and getter each fields as needed	

2 Package entity.game

2.1 Class type1 extends Tower

เป็น Tower เริ่มต้นแบบแรกในเกม

2.1.1 Field

เหมือนกับ Tower

2.1.2 Constructor

+ type1 (int speedatk , int attack, int r , int x , int y)	สร้าง type1 จาก speedatk , attack , r , x และ y และกำหนดให้ r เท่ากับ 2 กับ กำหนด type เป็น BulletsType.NORMAL
--	--

+ type1 (type1 other)	เป็นการ copy Constructor
-------------------------	--------------------------

2.1.3 Method

+ void attack ()	ทำการหา Monster ที่ใกล้ที่สุดและอยู่ภายในขอบเขตของ Tower โดยใช้ method findMonster () ถ้าเป็น null ไม่ต้องทำอะไร ถ้าไม่ ให้สร้าง Bullets ขึ้นมา
+ Optional<Tower> get_upgrade_lsh ()	คืน Tower ใหม่ที่เหมือน Tower อันเก่า แต่ทำการเพิ่มราคาของ Tower เท่ากับราคาที่ upgrade สร้าง Tower ใหม่ และถ้า Tower level = 1 จะทำการเพิ่ม speedatk อีก 2 ถ้า level = 2 ให้ทำการเพิ่ม base attack อีก 200 และ ถ้า level = 3 Tower นั้นจะกลายเป็น Tower Fire ซึ่งจะเพิ่ม base attack อีก 500
+ Optional<Tower> get_upgrade_rsh ()	คืน Tower ใหม่ที่เหมือน Tower อันเก่า แต่ทำการเพิ่มราคาของ Tower เท่ากับราคาที่ upgrade สร้าง Tower ใหม่ และถ้า Tower level = 1 จะทำการเพิ่ม r อีก 0.5 ถ้า level = 2 ให้ทำการเพิ่ม r อีก 0.5 และ ถ้า level = 3 Tower นั้นจะกลายเป็น Tower Ice ซึ่งจะเพิ่ม speedattack อีก 4
+ Image getSprite ()	Return Sprites.TowerType1
+ int upgradePrice_lsh ()	เชื่อว่า level นั้นควรมีราคา upgrade ฝั่งซ้ายเท่ากับเท่าไร โดยที่ level = 1 จะคืนค่า 100 level = 2 จะคืนค่า 200 และ level = 3 จะคืนค่า 450
+ int upgradePrice_rsh ()	เชื่อว่า level นั้นควรมีราคา upgrade ฝั่งขวาเท่ากับเท่าไร โดยที่ level = 1 จะคืนค่า 100 level = 2 จะคืนค่า 200 และ level = 3 จะคืนค่า 400

2.2 Class type2 extends Tower

เป็น Tower เริ่มต้นแบบแรกในเกม

2.1.1 Field

เหมือนกับ Tower

2.1.2 Constructor

+ type2 (int speedatk , int	สร้าง type1 จาก speedatk , attack , r , x และ y และกำหนดให้
------------------------------	---

attack, int r , int x , int y)	r เท่ากับ 2 กับ กำหนด type เป็น BulletsType.NORMAL
+ type2 (type2 other)	เป็นการ copy Constructor

2.1.3 Method

+ void attack ()	ทำการหา Monster ที่ใกล้ที่สุดและอยู่ในขอบเขตของ Tower โดยใช้ method findMonster () ถ้าเป็น null ไม่ต้องทำอะไร ถ้าไม่ ให้สร้าง Bullets ขึ้นมา ทั้งนี้ถ้า level tower มากกว่า 2 จะไม่สามารถทำการโจมตีได้
+ Optional<Tower> get_upgrade_lsh ()	ทำการเพิ่มราคาของ Monster เท่ากับราคาที่ upgrade สร้าง Tower ใหม่ และถ้า Tower level = 1 จะทำการเพิ่ม speedatk อีก 2 ถ้า level = 2 ให้ทำการเพิ่ม r อีก 0.25 และ ถ้า level = 3 Tower นั้นจะกลายเป็น Tower Farm
+ Optional<Tower> get_upgrade_rsh ()	ทำการเพิ่มราคาของ Monster เท่ากับราคาที่ upgrade สร้าง Tower ใหม่ และถ้า Tower level = 1 จะทำการเพิ่ม attack อีก 200 ถ้า level = 2 ให้ทำการเพิ่ม r อีก 0.25 และ ถ้า level = 3 Tower นั้นจะกลายเป็น Tower Strength
+ void sell ()	จะทำการลบ Tower ออกไปและเพิ่มเงินให้ผู้เล่นเท่ากับราคาของ Tower นั้นหาร 10 โดยต้องตรวจว่าถ้า Tower เป็นประเภท Farm หรือ Strength ต้องลบ Tower นั้นจาก ArrayList ที่เก็บ Farm หรือ Strength นั้นด้วย
+ Image getSprite ()	Return Sprites.TowerType2
+ int upgradePrice_lsh ()	เชื่อว่า level นั้นควรมีราคา upgrade ฝั่งซ้ายเท่ากับเท่าไร โดยที่ level = 1 จะคืนค่า 150 level = 2 จะคืนค่า 300 และ level = 3 จะคืนค่า 600
+ int upgradePrice_rsh ()	เชื่อว่า level นั้นควรมีราคา upgrade ฝั่งขวาเท่ากับเท่าไร โดยที่ level = 1 จะคืนค่า 150 level = 2 จะคืนค่า 250 และ level = 3 จะคืนค่า 550

2.3 Class type3 extends Tower

เป็น Tower เริ่มต้นแบบแรกในเกม

2.1.1 Field

เหมือนกับ Tower

2.1.2 Constructor

+ type3 (int speedatk , int attack, int r , int x , int y)	สร้าง type1 จาก speedatk , attack , r , x และ y และกำหนดให้ r เท่ากับ 2 กับ กำหนด type เป็น BulletsType.NORMAL
+ type3 (type3 other)	เป็นการ copy Constructor

2.1.3 Method

+ void attack ()	ทำการหา Monster ที่ใกล้ที่สุดและอยู่ภายในขอบเขตของ Tower โดยใช้ method findMonster () ถ้าเป็น null ไม่ต้องทำอะไร ถ้าไม่ ให้สร้าง Bullets ขึ้นมา
+ Optional<Tower> get_upgrade_lsh ()	ทำการเพิ่มราคาของ Monster เท่ากับราคาที่ upgrade สร้าง Tower ใหม่ และถ้า Tower level = 1 จะทำการเพิ่ม speedatk อีก 0.5 ถ้า level = 2 ให้ทำการเพิ่ม base attack อีก 200 และ ถ้า level = 3 Tower นั้นจะกลายเป็น Tower Laser ซึ่งเพิ่ม base attack อีก 900
+ Optional<Tower> get_upgrade_rsh ()	ทำการเพิ่มราคาของ Monster เท่ากับราคาที่ upgrade สร้าง Tower ใหม่ และถ้า Tower level = 1 จะทำการเพิ่ม r อีก 0.25 ถ้า level = 2 ให้ทำการเพิ่ม speedatk อีก 0.5 และ ถ้า level = 3 Tower นั้นจะกลายเป็น Tower Strength ซึ่งเพิ่ม base atk อีก 400
+ void sell ()	จะทำการลบ Tower ออกไปและเพิ่มเงินให้ผู้เล่นเท่ากับราคาของ Tower นั้นหาร 10 โดยต้องตรวจว่าถ้า Tower เป็นประเภท Farm หรือ Strength ต้องลบ Tower นั้นจาก ArrayList ที่เก็บ Farm หรือ Strength นั้นด้วย
+ Image getSprite ()	Return Sprites.TowerType3
+ int upgradePrice_lsh ()	เช็คว่า level นั้นควรมีราคา upgrade ฝั่งซ้ายเท่ากับเท่าไร โดยที่ level = 1 จะคืนค่า 200 level = 2 จะคืนค่า 250 และ level = 3 จะคืนค่า 450
+ int upgradePrice_rsh ()	เช็คว่า level นั้นควรมีราคา upgrade ฝั่งขวาเท่ากับเท่าไร โดยที่

	level = 1 จะ কিনค่า 100 level = 2 จะ কিনค่า 300 และ level = 3 จะ কিনค่า 450
--	---

2.4 Class Fire extends type1

Level max ของ type1

2.4.1 Field

เหมือนกับ Tower

2.4.2 Constructor

+ Fire (int speedatk , int attack , int price , int x , int y)	สร้าง Fire จาก speedatk , attack , price , x และ y และกำหนด type ให้เป็น BulletsType.BURN
--	---

2.4.3 Method

+ Optional<Tower> get_upgrade_lsh ()	কিনค่า Tower เมื่อ upgrade ฝั่งซ้าย
+ Optional<Tower> get_upgrade_rsh ()	কিনค่า Tower เมื่อ upgrade ฝั่งขวา
+ Image getSprite ()	Return Sprites.TowerFire

2.5 Class Ice extends type1

Level max ของ type1

2.5.1 Field

เหมือนกับ Tower

2.5.2 Constructor

+ Ice (int speedatk , int attack , int price , int x , int y)	สร้าง Ice จาก speedatk , attack , price , x และ y และกำหนด type ให้เป็น BulletsType.FREEZE
---	--

2.5.3 Method

+ Optional<Tower> get_upgrade_lsh ()	কিনค่า Tower เมื่อ upgrade ฝั่งซ้าย
+ Optional<Tower>	কিনค่า Tower เมื่อ upgrade ฝั่งขวา

get_upgrade_rsh ()	
+ Image getSprite ()	Return Sprites.TowerIce

2.6 Class Farm extends type2

Level max ของ type2

2.6.1 Field

เหมือนกับ Tower

2.6.2 Constructor

+ Farm (int speedatk , int attack , int price , int x , int y)	สร้าง Farm จาก speedatk , attack , price , x และ y และกำหนด type ให้เป็น null เพิ่ม r อีก 1 และเก็บ Farm ใน Simulation.getFarm ()
--	--

2.6.3 Method

+ Image getSprite ()	Return Sprites.TowerFarm
+ void attack ()	return

2.7 Class Strength extends type2

Level max ของ type2

2.7.1 Field

เหมือนกับ Tower

2.7.2 Constructor

+ Strength (int speedatk , int attack , int price , int x , int y)	สร้าง Strength จาก speedatk , attack , price , x และ y และกำหนด type ให้เป็น null เพิ่ม r อีก 1 และเก็บ Farm ใน Simulation.getFarm ()
--	--

2.7.3 Method

+ Image getSprite ()	Return Sprites.TowerStrength
+ void attack ()	return

2.8 Class Laser extends type3

Level max ของ type3

2.8.1 Field

เหมือนกับ Tower

2.8.2 Constructor

+ Laser (int speedatk , int attack , int price , int x , int y)	สร้าง Laser จาก speedatk , attack , price , x และ y และกำหนด type ให้เป็น BulletsType.PIERCE
---	--

2.8.3 Method

+ Image getSprite ()	Return Sprites.TowerLaser
-----------------------	---------------------------

2.9 Class Boom extends type3

Level max ของ type3

2.9.1 Field

เหมือนกับ Tower

2.9.2 Constructor

+ Boom (int speedatk , int attack , int price , int x , int y)	สร้าง Boom จาก speedatk , attack , price , x และ y และกำหนด type ให้เป็น null
--	---

2.9.3 Method

+ Image getSprite ()	Return Sprites.TowerBoom
+ void attack ()	ทำการลด hp ของ monster ทุกตัวที่อยู่ในขอบเขตของ Tower เท่ากับ 100

2.10 Class soldier extends Monster

เป็น Monster แบบปกติ

2.10.1 Field

เหมือนกับ Monster

2.10.2 Constructor

+ soldier (int hp , int speed)	สร้าง soldier จาก hp และ speed
+ soldier ()	สร้าง soldier hp=1000 และ speed=1

2.10.3 Method

2.11 Class speedsoldier extends soldier

เป็น soldier ที่เดินเร็วขึ้น

2.11.1 Field

เหมือนกับ soldier

2.11.2 Constructor

+ speedsoldier ()	สร้าง speedsoldier hp=1500 และ speed=2 และ กำหนดค่า dlife เป็น 2
-------------------	--

2.12 Class boss extends Monster

เป็น boss ของ Monster

2.12.1 Field

เหมือนกับ Monster

2.12.2 Constructor

+ boss ()	สร้าง boss hp=10000 และ speed=0.5 และ กำหนดค่า dlife เป็น 50
-----------	--

3 Package logic

3.1 Class Simulation

เป็นการคำนวณของ เงิน และ ค่าชีวิตของผู้เล่น รวมทั้งยังเก็บ Tower บางประเภท

3.1.1 Field

- <u>int money</u>	เก็บเงินของผู้เล่น โดยตอนเริ่มเกมให้มียกเท่ากับ 500
- <u>int lifepoint</u>	เก็บเลือดของผู้เล่น โดยตอนเริ่มเกมให้มียกเท่ากับ 150
- <u>int round</u>	เก็บจำนวน round ที่ผ่านมา โดยตอนเริ่มเกมให้มียกเท่ากับ 1
- <u>ArrayList<Strength> strength</u>	เก็บ Tower ที่มี class Strength ทั้งหมดของเกม
- <u>ArrayList<Farm> farm</u>	เก็บ Tower ที่มี class Farm ทั้งหมดของเกม

3.1.2 Constructor

3.1.3 Method

+ <u>void decreaseMoney (int price)</u>	ลดเงินของผู้เล่นไปเท่ากับ price และถ้า price มีค่ามากกว่าเงินที่มีอยู่ให้เงินผู้เล่นเท่ากับ 0
+ <u>void increaseMoney (int price)</u>	เพิ่มเงินของผู้เล่นให้อีก price ถ้า price มีค่าน้อยกว่า 0 ไม่ต้องทำอะไร
+ Boolean isLose ()	ถ้า lifepoint น้อยกว่าเท่ากับ 0 ให้คืนค่า false ถ้าไม่คืนค่า true
+ setters and getter each fields as needed	

3.2 Class Towers

3.2.1 Field

- final ArrayList<ArrayList<Tower>> arrayGrid	สร้าง ArrayList<ArrayList<Tower>>
- Pair<Integer,Integer> selectedPosition	กำหนดให้เป็น null

3.2.2 Constructor

+ Towers ()	เรียก method makeMap ()
--------------	--------------------------

3.2.3 Method

+ void makeMap ()	สร้าง ArrayList 2 มิติ ขนาด 50 x 50 ช่างในเก็บค่า null ทั้งหมด
+ void setTower (int x , int y ,Tower tower)	ใส่ Tower tower ในตำแหน่ง xy ใน arrayGrid แล้วเรียกใช้ tower.recalculateStrengthBuff กับทุกๆ tower ที่มีอยู่
+ void deleteTower (int x ,int y)	ลบ Tower ใน arrayGrid โดยเปลี่ยน ค่าในตำแหน่ง xy ให้เป็น null แล้วเรียกใช้ tower.recalculateStrengthBuff กับทุกๆ tower ที่มีอยู่
+ void Optional<Tower> getTower (int x, int y)	คืนค่า Tower ที่ตำแหน่ง xy
+ ArrayList<Tower> getRow (int y)	คืนค่า ArrayList<Tower> จากตำแหน่ง y ใน arrayGrid

)	
+ ArrayList<ArrayList<Tower>> asArrayList ()	คืนค่า arrayGrid
+ void iterateTower (BiConsumer<Pair<Integer,Integer>, Tower> f)	ทำการรัน f กับ Tower ทุกตัวที่มีอยู่ใน Map
+ Boolean select (int x , int y)	ทำการรันเมื่อผู้เล่นกดเลือก Tower บน Map ถ้าเป็น Tower คืนค่า true ถ้าไม่มี Tower อยู่คืนค่า false
+ Optional<Pair<Integer, Integer>> getSelectedPosition ()	คืนค่า ว่าผู้เล่นเลือกตำแหน่งไหนอยู่

4 Package core.timing

4.1 Class FpsCounter implements Draw, Tick

วัด Fps ตลอด 5 วินาทีและแสดงผลบน Canvas

4.1.1 Field

- Interval fpsTimer	เป็น Timer ที่ run ทุก 5 วินาที เมื่อครบ 5 วินาทีจะแสดงผล fps บนหน้าจอและ reset frameCount กลับเป็น 0
- double lastFps	เก็บ fps ล่าสุด
- int frameCount	เก็บจำนวน frame ตั้งแต่การ Reset ที่แล้ว
- int double x	ตำแหน่ง x ในหน่วย px ที่แสดงบนหน้าจอ
- int double y	ตำแหน่ง y ในหน่วย px ที่แสดงบนหน้าจอ

4.1.2 Constructor

+ FpsCounter (double x , double y)	สร้าง FpsCounter ที่ตำแหน่ง x,y
--------------------------------------	---------------------------------

4.1.3 Method

+ void tick (double dt)	Game จะ Run function นี้ ทุก frame เพิ่ม frameCount มา 1
---------------------------	---

4.2 Class Interval

Timer ที่ใช้ run function ทุกๆ period วินาที

ตัวนี้ไม่ได้ผูกกับนาฬิกาภายนอกใดๆ แต่จะใช้การรับ dt คือเวลา(ในหน่วยวินาที)ตั้งแต่ตอน

รับ dt ถึงปัจจุบัน

เมื่อนำ dt มารวมกันก็จะได้ระยะเวลารวม หากรวมกันเกิน period ก็แสดงว่าเวลาสะสมกันผ่าน

มาเกินกำหนดแล้วจึงต้องทำการ run function และ reset ระยะเวลารวมกลับเป็น 0

4.2.1 Field

- double period	ระยะเวลาระหว่างแต่ละการ run
- double accTime	Accumulated time เก็บระยะเวลารวมตั้งแต่การ run ครั้งล่าสุด

4.2.2 Constructor

+ Interval (double period)	สร้าง Interval จาก period คือระยะเวลาระหว่างการ run
------------------------------	---

4.2.3 Method

+ void resetTime ()	ทำให้ accTime เป็น 0
+ Boolean addTime (double dt)	เพิ่ม dt รวมไปใน accTime, return ว่า accTime เกิน period หรือยัง
+ void tick (double dt , Consumer<Double> runner)	รับ dt กับ function ที่จะ run แล้วจะนำ dt รวมเข้าไปในระยะเวลา รวมด้วย addTime หากรวมแล้วเกินกำหนดก็จะ run runner และ resetTime() หาก dt ถูกต้องผลลัพธ์ที่ได้คือ runner จะทำงานทุกๆ period วินาที
+ setters and getter each fields as needed	

6 Package level

6.1 Abstract class Level

แสดงถึงด่านๆหนึ่งในเกม

6.1.1 Field

6.1.2 Constructor

6.1.3 Method

+ abstract <i>TileGrid</i> <i>getTileGrid</i> ()	ให้ <i>TileGrid</i> ของด่าน
+ abstract <i>Spawner</i> <i>nextSpawner</i> ()	ให้ <i>Spawner</i> ของ round ถัดมา

6.2 Class Level1 extends Level

6.2.1 Field

- <i>TileGrid</i> <i>tileGrid</i>	เก็บ <i>TileGrid</i> ของ level 1
- <i>Track</i> <i>sampleTrack</i>	เก็บ <i>Track</i> คือเส้นทางของ Monster ใน Level 1

6.2.2 Constructor

+ <i>Level1</i> ()	สร้าง <i>tileGrid</i> จาก folder <i>map/level1</i> และ <i>sampleTrack</i> (Hardcode)
---------------------	--

6.2.3 Method

+ <i>TileGrid</i> <i>getTileGrid</i> ()	Getter ของ <i>tileGrid</i>
+ <i>Spawner</i> <i>nextSpawner</i> ()	Return <i>Spawner</i> ของ round ถัดไปใน Level 1

6.3 Class Track

เส้นทางที่ Monster เคลื่อนที่ เก็บเป็น array ของ *Point2D*

เริ่มมา Monster จะไปอยู่ที่ *Point2D* แรกใน array แล้วเคลื่อนที่เป็นเส้นตรงหาเป้าหมายคือ

Point2D อันถัดมา เมื่อมันเคลื่อนถึงเป้าหมายมันจะเคลื่อนที่ไปเป้าหมายถัดไปคือ *Point2D* อัน

ถัดมาอีก ไปเรื่อยๆจนครบทุกจุดใน array

เนื่องจาก logic ไปอยู่ใน Monster หมด, Class นี้ใช้เป็นเหมือน New type pattern สำหรับ

Point2D[] เราจึงเลือกให้ *path* เป็น public ไปเลย

6.3.1 Field

+ <i>Point2D</i> [] <i>path</i>	เก็บ array ของพิกัด ที่ Monster ใช้ในการเคลื่อนที่
----------------------------------	--

6.3.2 Constructor

+ Track (Point2D[] path)	สร้าง Track จาก array ของพิกัด
-----------------------------	--------------------------------

6.3.3 Method

+ String toString ()	เป็น path.toString()
-----------------------	----------------------

7 Package level.spawner

7.1 Class PeriodicSpawner extends Spawner

Spawner ที่ผลิต Monster เป็นช่วงๆ

7.1.1 Field

- final Interval timer	Timer เพื่อควบคุมเวลาในการ spawn Monster
- final Supplier<Monster> factory	Function ที่ใช้ผลิต Monster
- int limit	จำนวน Monster ทั้งหมดที่จะผลิต ถ้าเป็น -1 จะผลิตไม่รู้จบ
- int spawnCount	จำนวน Monster ที่ผลิตมาแล้ว

7.1.2 Constructor

+ PeriodicSpawner (Supplier<Monster> factory , double interval)	สร้าง PeriodicSpawner จาก factory กับระยะเวลาระหว่างการ ผลิต Monster โดยผลิตไม่รู้จบ
+ PeriodicSpawner (Supplier<Monster> factory, double interval , int limit)	สร้าง PeriodicSpawner จาก factory กับระยะเวลาระหว่างการ ผลิต Monster และจำนวนการผลิตทั้งหมด

7.1.3 Method

+ Boolean isDone ()	ให้ว่าผลิตครบจำนวนแล้วหรือยัง (spawnCount >= limit), หาก กำลังผลิตแบบไม่รู้จบ (limit = -1) จะให้ค่า false ตลอด
+ void tick (double dt)	ส่งต่อ dt ไปใช้ใน timer เพื่อผลิต Monster จาก factory เป็นช่วงๆ ทุกครั้งที่ผลิตจะเพิ่ม spawnCount มา 1
+ int getSpawnCount ()	Getter ของ spawnCount

7.2 Class SequentialSpawner extends Spawner

Spawner ที่รวมเก็บ Spawner หลายๆตัว โดยจะในการผลิต Monster จะเริ่มผลิตจาก Spawner ตัวแรกจนหมดแล้วไปตัวสอง ตัวสาม ไปเรื่อยๆจนครบทุกตัว
(Concept คล้ายๆกับ [IteratorChain](#) ในโลกของ iterator)

7.2.1 Field

- final Spawner[] spawners	เก็บ array ของ spawner
- int i	เก็บว่าตอนนี้ผลิตจาก spawner index เท่าไหร่อยู่

7.2.2 Constructor

+ SequentialSpawner (Spawner[] spawners)	Initialize SequentialSpawner โดยเก็บ spawners ไปใช้ในกระบวนการผลิต ตั้งให้ทุกตัวใน spawners เมื่อผลิต monster ออกมาจะส่งมาให้ตัวนี้ นี้ทำเสมือนว่าตัวนี้ผลิตเอง
--	---

7.2.3 Method

+ tick (double dt)	ส่งต่อไปให้ spawner ที่ index i ถ้า spawner นั้นผลิตครบแล้วจะเพิ่ม i มา 1 เพื่อให้ครั้งหน้าใช้ spawner ตัวถัดไป หากผลิตเสร็จหมดแล้ว (ใช้ isDone() ตรวจสอบ) method นี้จะไม่ทำอะไร
+ boolean isDone ()	ให้ว่าผลิตเสร็จยัง ตัวนี้จะผลิตเสร็จเมื่อ i >= spawners.size()

7.3 Abstract Class Spawner implements Tick

ใช้กำหนดการเกิดของ Monster, เมื่อ tick ทำงานอย่างต่อเนื่องต่อเนื่อง มันจะผลิต Monster มาเรื่อยๆ

การผลิต Monster ไม่ได้ spawn Monster ออกมาใน Game ตรงๆ แต่จะส่ง Monster ที่ผลิตมาให้กับ [Consumer](#) ซึ่งมันอาจนำ monster ไป spawn ใน Game จริงๆ
หรืออาจนำไปทำอย่างอื่นก็ได้

7.3.1 Field

- Consumer<Monster> onSpawn	เมื่อ spawner ผลิต monster มาจะส่งให้ consumer นี้
- Track track	เก็บ track ที่ monster ที่ spawner ผลิตมาจะใช้

7.3.2 Constructor

7.3.3 Method

+ void setOnSpawn (Consumer<Monster> onSpawn)	Setter ของ onSpawn
+ Spawner setTrack (Track track)	Setter ของ track และคืนค่า this (เพื่อ method chaining)
+ abstract boolean isDone ()	ให้ว่า Spawner ผลิตเสร็จหมดยัง

7.4 Class EmptySpawner extends Spawner

Spawner ที่ไม่ผลิต Monster อะไรเลย มีหน้าที่ไว้ถ่วงเวลา

7.4.1 Field

- double countdown	เก็บระยะเวลา(วินาที)ก่อนที่จะถือว่าตัวนี้ผลิตเสร็จ
--------------------	--

7.4.2 Constructor

+ EmptySpawner (double delay)	ตั้ง countdown = delay
--------------------------------	------------------------

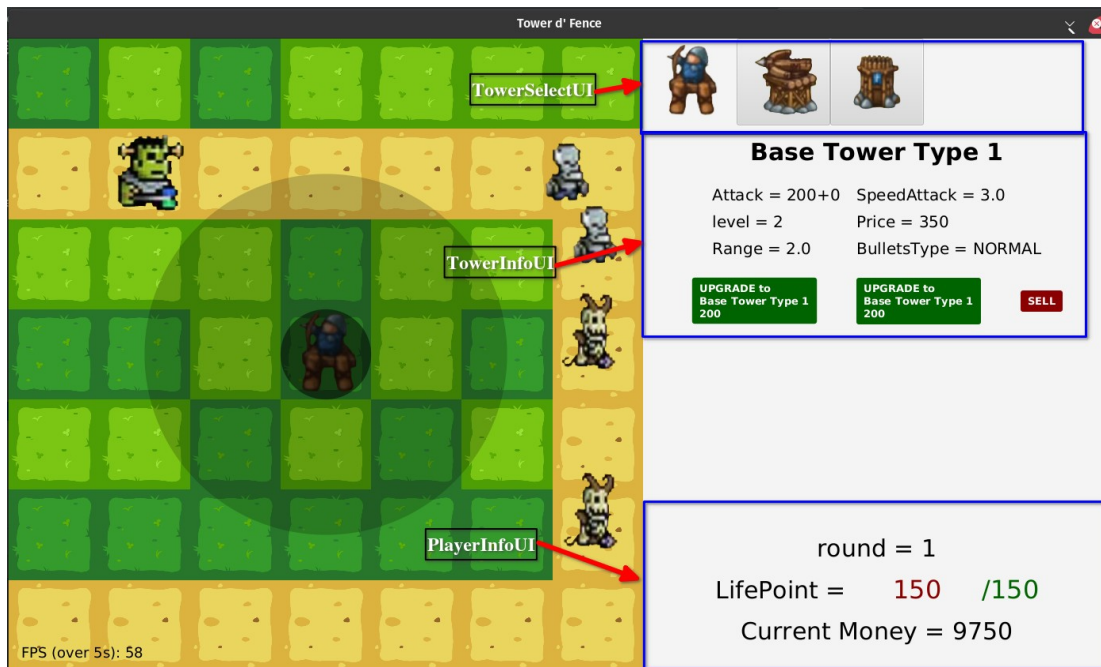
7.2.3 Method

+ tick (double dt)	ลด countdown ไปเป็นจำนวน dt
+ boolean isDone ()	ให้ว่าผลิตเสร็จยัง ตัวนี้จะผลิตเสร็จเมื่อ countdown <= 0

8 Package ui

8.1 Class Sidebar extends VBox

UI แถบด้านขวา



8.1.1 Field

- final TowerSelectUI towerSelectUI ()	UI ของแถบที่ใช้เลือก tower ไปวาง
- final TowerInfoUI towerInfoUI	UI ของส่วนที่แสดงข้อมูลของ tower ที่เลือกอยู่
- final PlayerInfoUI playerInfoUI	UI ของส่วนที่แสดงข้อมูลของผู้เล่น

8.1.2 Constructor

+ Sidebar ()	Initialize field ทั้งหมดด้วย default constructor of appropriate class
---------------	---

8.1.3 Method

+ void refresh()	เรียก refresh method ของ towerInfoUI และ playerInfoUI
+ Getters	

8.2 Class TowerButton extends Button

ปุ่มเลือก tower ที่จะไปสร้าง



8.2.1 Field

- ImageView imageView	ใช้แสดงภาพของ TowerButton นี้
- BiFunction<Integer, Integer, Tower> factory	Function รับตำแหน่ง x, y ที่ผู้เล่นเลือกจะวาง tower แล้วมันจะผลิต tower ออกมา

8.2.2 Constructor

+ TowerButton (BiFunction<Integer, Integer, Tower> factory , Image sprite) +TowerButton(BiFunction<Integer, Integer, Tower> factory)	สร้าง TowerButton จาก factory กับ sprite sprite จะนำเอาไปใช้สร้าง imageView หา sprite จาก tower.getIconSprite() โดย tower สร้างมาจากการใส่พิกัด 0,0 ลงไปใน factory (tower = factory.apply(0, 0)) แล้วนำ factory กับ sprite ไปใช้กับ constructor ด้านบน
--	---

8.2.3 Method

+ void highlight ()	ตั้ง UI ตอนปุ่มนี้โดนเลือก
+ void unhighlight ()	ตั้ง UI ตอนปุ่มไม่โดนเลือก
+ Getters	

8.3 Class TowerInfoUI extend VBox

แสดงรายละเอียดของ tower ที่ถูกเลือก

8.3.1 Field

- GridPane grid	เก็บค่า properties ต่าง ๆ
- Text mainText	แสดงค่า ชื่อ ของ Tower
- Text attack	แสดง attack ของ Tower
- Text price	แสดง price ของ Tower
- Text speedattack	แสดง speedattack ของ Tower
- Text bullettype	แสดง type ของ Tower
- Text range	แสดง ขอบเขตของ Tower
- Text level	แสดง level ของ Tower
- Button sellBtn	ปุ่มสำหรับการขาย Tower
- Button upgradeLeft	ปุ่มที่ไว้ upgrade Tower ฝั่งซ้าย
- Button upgradeRight	ปุ่มที่ไว้ upgrade Tower ฝั่งขวา
- int towerX	เก็บตำแหน่ง x ของ tower ที่ผู้เล่นเลือก
- int towerY	เก็บตำแหน่ง y ของ tower ที่ผู้เล่นเลือก

8.3.2 Constructor

+ TowerInfoUI ()	Default Initialize field ทั้งหมด
-------------------	----------------------------------

8.3.3 Method

+ void seeTower (int x , int y)	ทำการปรับค่า field ทั้งหมดให้ตรงกับ Tower ที่อยู่ ณ ตำแหน่ง x,y
+ void refresh ()	นำข้อมูลจาก Tower มาแสดงผลอีกรอบ หากผู้เล่นเลือก tower อยู่
+ void unseeTower ()	หยุดการแสดงผลข้อมูลของ tower

8.4 Class TowerSelectUI extends TilePane

8.4.1 Field

- ObservableList<TowerButton> towerButtons	เก็บ TowerButton ทั้งหมด
---	--------------------------

- Optional<Integer> selectingIndex	เก็บ index ของ TowerButton ที่เลือกอยู่ ถ้ามี
---------------------------------------	---

8.4.2 Constructor

8.4.3 Method

+ addTowerButton (TowerButton towerButton)	เพิ่ม Tower Button
+ void habdleButtonClick (int I)	ใช้เมื่อผู้เล่นกดคลิกที่ TowerButton ตำแหน่ง i
+ void deselect ()	หยุดเลือก TowerButton ทั้งหมด
+ void select (int I)	เลือก TowerButton
+ Optional<TowerButton> getSelected ()	Return TowerButton ที่เลือกอยู่

8.3 Class PlayerInfoUI extend VBox

แสดงรายละเอียดของผู้เล่น

8.3.1 Field

- Text round	แสดงจำนวนรอบของเกม
- Text myhp	แสดง lifepoint ของผู้เล่น
- Text maxhp	แสดง lifepoint สูงสุดของผู้เล่น
- Text money	แสดงเงินของผู้เล่น

8.3.2 Constructor

+ TowerInfoUI ()	Default Initialize field ทั้งหมด
-------------------	----------------------------------

8.3.3 Method

+ void refresh ()	ดึงข้อมูลล่าสุดจากมาแสดงผล
--------------------	----------------------------

9 Package utils

9.1 Class InputUtils

9.1.1 + *MouseTrack* (Static inner class)

Track ตำแหน่ง mouse ใน canvas ด้วย MouseMoved event

9.1.1.1 Field

- Point2D gridPos	เก็บตำแหน่ง mouse ล่าสุด (In game coordinate)
-------------------	---

9.1.1.2 Constructor

+ MouseTrack()	Initialize gridPos to 0,0
----------------	---------------------------

9.1.1.3 Method

- void updatePixelPos(double x, double y) + Getters	Update gridPos ด้วยตำแหน่งใหม่ (x, y) พิกัด (x, y) เป็น pixel/screen coordinate แต่ gridPos เป็น grid/game coordinate จึงต้องทำการแปลงก่อน
--	--

9.1.2 Field

+ final MouseTrack mouse	ไว้ track ตำแหน่ง mouse
--------------------------	-------------------------

9.1.3 Constructor

+ InputUtils (Canvas canvas)	ตั้งให้เมื่อใน Canvas เกิด MouseMoved event ขึ้น จะนำเอาตำแหน่ง mouse จาก event นั้นไป run mouse.updatePixelPos(x, y)
--------------------------------	---

9.2 Class MapLoader

Helper Class เพื่อใช้ load TileGrid จาก folder

9.2.1 Field

9.2.2 Method

+ <u>TileGrid loadMap (String levelName) throw IOException,URISyntaxException</u>	ใช้ ClassLoader.getResource หา folder ชื่อ levelName ใน folder map (หา map/{levelName}) ออกมาเป็น Path แล้วส่งต่อให้ loadMap(Path)
+ <u>TileGrid loadMap (Path mapFolder) throws IOException</u>	สร้าง TileGrid จากข้อมูลใน folder ณ mapFolder throw IOException หาก folder ไม่มีอยู่จริงหรือข้อมูลใน folder ใช้ไม่ได้

9.3 Class Sound

load และเก็บทุก Audio ในเกม, Audio ถูก load ใน static block

9.3.1 Field

+ <u>AudioClip TowerPlace</u>	Audio ตอนวาง tower ใหม่
+ <u>AudioClip BulletHit</u>	Audio ตอน monster โดน bullet
+ <u>AudioClip Hurt1</u>	Audio ตอน monster เข้าไปลดเลือดผู้เล่น
+ <u>AudioClip Hurt2</u>	Audio ตอน monster เข้าไปลดเลือดผู้เล่นอีกอันหนึ่ง

9.3.2 Constructor

9.3.3 Method

+ <u>AudioClip loadAudio (String filename) throws IOException</u>	ใช้ ClassLoader load file ใน folder audio ที่ชื่อไฟล์เป็น filename และแปลงมันเป็น AudioClip AudioClip ทุกตัวใน class นี้ load ผ่าน function นี้
---	--

9.4 Class Sprites

load และเก็บภาพทุก Image ในเกม, Image ถูก load ใน static block

9.4.1 Field

+ <u>Image GrassASprite</u>	Image ของหญ้าพื้นหลัง
+ <u>Image GrassBSprite</u>	Image ของหญ้าพื้นหลังอีก variant
+ <u>Image SandSprite</u>	Image ของพื้นหลังทราย
+ <u>Image TowerType1</u>	Image ของ Type1 tower
+ <u>Image TowerType2</u>	Image ของ Type2 tower
+ <u>Image TowerType3</u>	Image ของ Type3 tower
+ <u>Image TowerFire</u>	Image ของ Fire tower
+ <u>Image TowerIce</u>	Image ของ Ice tower
+ <u>Image TowerStrength</u>	Image ของ Strength tower
+ <u>Image TowerFarm</u>	Image ของ Farm tower
+ <u>Image TowerLaser</u>	Image ของ Laser tower
+ <u>Image TowerBoom</u>	Image ของ Boom tower
- <u>final HashMap<String , Image> imageByName</u>	Hashmap ที่ map ชื่อไฟล์ไปยัง Image สำหรับทุกภาพที่เคยใช้ loadImage ทำมาแล้ว ใช้ป้องกัน load ภาพเดิมซ้ำซ้อน

9.4.2 Constructor

9.4.3 Method

+ <u>Image loadImage (String filename)</u>	ถ้า filename เคยมีใน imageByName อยู่แล้ว จะใช้ return getImage(filename) ไปเลย ถ้าไม่มีจะใช้ ClassLoader load file ใน folder sprite ที่ชื่อไฟล์เป็น filename และแปลงมันเป็น Image และใส่ชื่อไฟล์กับ Image ลงไปใน imageByName Image ทุกตัวใน class นี้ load ผ่าน function นี้
+ <u>Optional<Image> getImage (String filename)</u>	ให้ค่า Image ที่ตรงกับ filename หากเคยโหลด(ด้วย loadImage)มาแล้ว กลไกคือคืนค่าจาก imageByName เมื่อใช้ filename เป็น key

9.5 Class TileGrid implements Draw

เก็บข้อมูลเกี่ยวกับ Tile พื้นหลัง

ในขั้นตอนตอนอ่าน Tilemap จาก folder เช่นจาก map/level1, ใน folder นั้นจะ file อยู่ 2 อันชื่อ tile.tsv กับ tile_props.tsv

tile_props.tsv มีหน้าตาประมาณนี้

tile	sprite	tower_placeable
A	grass_tile_1	1
B	grass_tile_3	1
S	sand_tile	0

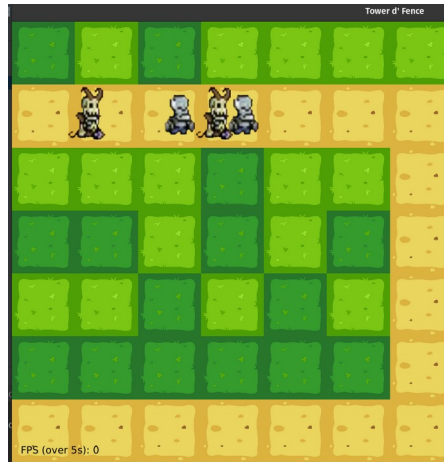
ในตัวอย่างนี้แถวแรกแสดงว่า tile ชื่อ A ใช้ภาพ sprite/grass_tile_1 และสามารถวาง tower ได้

tile.tsv มีหน้าตาประมาณนี้

A	B	A	B	B	B	B
S	S	S	S	S	S	S
B	B	B	A	B	B	S
A	A	B	A	B	A	S
B	B	A	B	A	B	S
A	A	A	A	A	A	S
S	S	S	S	S	S	S

เป็นตาราง 2 มิติแสดงว่าแต่ละตำแหน่งเป็น tile ชื่ออะไรบ้าง

ในตัวอย่างนี้แสดงว่า tile มุมล่างซ้ายสุดเป็น S เป็นพื้นที่วาง tower ไม่ได้



9.5.1 Field

- Image[][] sprites	2D array เก็บภาพพื้นหลังในแต่ละ tile
- Boolean[][] towerPlaceable	2D array เก็บว่าแต่ละ tile เป็นชนิดที่วาง tower ได้ไหม

9.5.2 Constructor

+ TileGrid (String[][] tile , String[][] tile_prop)	สร้าง TileGrid จาก 2D array of string ที่อ่านมาจาก tile.tsv และ tile_props.tsv
---	--

9.5.3 Method

+ int getIndexWidth ()	จำนวน Tile ใน 1 แถว, ความกว้างเกม
+ int getIndexHeight ()	จำนวน Tile ใน 1 column, ความสูงเกม
+ void draw (GraphicsContext gc, double dt)	ใช้ sprites วาดพื้นหลัง
+ Boolean isTowerPlaceable (int x, int y)	หาว่า Tile ที่ตำแหน่ง (x,y) เป็นชนิดที่วาง tower ได้ไหม

9.6 Class TilesProp

เก็บข้อมูลสมบัติของแต่ละชนิดของ tile (แปลงมาจาก tile_prop.tsv)

9.6.1 Field

- final HashMap<String, Image> spritesMap	Hashmap เชื่อมชื่อ tile ไปยังภาพ
- final HashMap<String, Boolean> towerPlaceable	Hashmap เชื่อมชื่อ tile ไปว่าสามารถวาง tower ได้ไหม

9.6.2 Constructor

+ TilesProp (String[][] tilePropSpec)	สร้าง TileProp จาก 2D array of string ที่อ่านมา จาก tile_props.tsv
--	---

9.6.3 Method

+ getter of each fields	
-------------------------	--

9.7 Class Utils

class ช่วย รวม function เบ็ดเตล็ด

9.7.1 Fields

- <u>Point2D</u> GridDim	Cache ของ getGridPixelDimension()
--------------------------	-----------------------------------

9.7.2 Constructor

9.7.3 Method

+ <u>String[][]</u> parseTsv (Path path)	แปลงไฟล์ tsv เป็น 2D array ของ string
+ <u>Point2D</u> grid2pixel (double x , double y)	แปลงจาก grid/game coordinate เป็น pixel/screen coordinate
+ <u>Point2D</u> grid2pixel (Point2D pos)	แปลงจาก grid/game coordinate เป็น pixel/screen coordinate
+ <u>Point2D</u> pixel2grid (double x , double y)	แปลงจาก pixel/screen coordinate เป็น grid/game coordinate

+ <u>Point2D pixel2Grid (Point2D pos)</u>	แปลงจาก pixel/screen coordinate เป็น grid/game coordinate
+ <u>Point2D getGridPixelDimension ()</u>	หาว่าพื้นที่ในเกมหนึ่งช่อง (1x1 square in grid/game coordinate) มีขนาดกี่ pixel บนหน้าจอ (pixel/screen coordinate) เนื่องจากหน้าจอ resize ไม่ได้ คำนี้นี้จึงเป็นค่าคงที่ แต่คำนวณใน static block ไม่ได้เพราะต้องรอ Game โหลดก่อน เราจึงทำการ cache ค่าลงใน <u>GridDim</u> ในการเรียกใช้ครั้งแรก การเรียกใช้ครั้งต่อๆมาก็เพียง return <u>GridDim</u>
+ <u>Point2D pair2point (Pair<Integer, Integer> pair)</u>	แปลงจาก Integer Pair เป็น Point
+ <u>drawSpriteFlipped (GraphicsContext gc, Image img ,double x , double y ,double w, double h)</u>	วาด Image แบบกลับซ้ายขวา (horizontal flip) parameter หลังจาก gc เหมือน gc.drawImage

10 Package core

10.1 Interface Draw

แสดงถึงความสามารถว่าวาดรูปลงไปใน canvas ได้

10.1 Field

10.2 Constructor

10.3 Method

+ void draw (GraphicsContext gc ,double dt)	วาดลงไปใน Canvas ผ่าน gc เวลาระหว่าง frame นี้กับก่อนหน้า เป็น dt (หน่วยวินาที)
---	---

10.2 Class Game implements Draw , Tick

Class หลักที่ควบคุมและส่งต่อการ draw และ tick จาก game loop ไปยังทุกอย่างในเกม

10.2.1 Field


- Level currentLevel	เก็บ Level ในเกมตอนนี้
- Spawner activeSpawner	เก็บ Spawner ที่ผลิต Monster อยู่ในตานี้
- ArrayList<Monster> monsters	เก็บ monster ทุกตัวในเกม
- ArrayList<Bullets> bullets	เก็บ bullet ทุกตัวในเกม
- ArrayList<DmgInd> dmgInds	เก็บ damage indicator ทุกตัวในเกม
- Towers towers	เก็บและคุม tower ในเกม
- HashMap<Pair<Integer,Integer>, HashSet<Monster>> monstersMap	HashMap ที่โยงพิกัดของ tile ไปยัง set ของ monster ทุกตัวที่อยู่บน tile นั้น

10.2.2 Constructor

+ Game (Level level)	Set currentLevel เป็น level Set activeSpawner เป็น level.nextSpawner() และตั้งเมื่อมันผลิต monster ให้เพิ่ม monster นั้นเข้าไปในเกม Default initialize field ที่เหลือ
-----------------------	---

10.2.3 Method

+ void draw (GraphicsContext gc , double dt)	(วาดทุกอย่างในเกม) วาด currentLevel.getTileGrid และ bullet, monster, damageIndicator ทุกตัวในเกม ผ่าน Draw.draw และเรียกใช้ drawTower และ drawTowerPlacement
- void drawTower (GraphicContext gc , double dt)	วาด tower ทุกอันในเกม โดยเรียกใช้ Tower.draw อย่างเหมาะสม และหากเลือก tower อยู่ให้วาด overlay ของ tower ที่ถูกเลือก โดยเรียกใช้ Tower.drawOverlay
- void drawTowerPlacement (GraphicsContext gc)	หาก tower button ใน Main.Sidebar.towerSelectUI โดนเลือกอยู่ ให้วาดภาพ tower ที่เกี่ยวข้องและโปร่งใส 50% ตรงตำแหน่ง tile ที่ mouse อยู่

	
+ void tick (double dt)	<p>(run logic ทุกอย่างในเกม)</p> <ol style="list-style-type: none"> 1. หาก activeSpawner ผลิตเสร็จแล้ว (Spawner::isDone) จะเป็นการจบหนึ่ง turn, ให้เรียกใช้ onTurnEnd() 2. Tick activeSpawner และ bullets, damageIndication ทุกตัว ผ่าน Tick::tick 3. เรียกใช้ tickTower กับ tickMonster <p>ตอนจบลบ entity ที่โดนทำลายไปแล้ว (Entity::isDestroyed) จะถูกลบออกจาก monsters, bullets, dmgInds ทั้งหมด</p>
+ void onTurnEnd()	<ol style="list-style-type: none"> 1. Set activeSpawner เป็น level.nextSpawner() และตั้งเมื่อมันผลิต monster ให้เพิ่ม monster นั้นเข้าไปในเกม 2. ทุกๆ Farm Tower จะเพิ่มเงินผู้เล่นมาอันละ 150 3. เรียกใช้ Simulation.nextRound();
- void tickMonster (double dt)	Tick Monster ทุกตัวในเกม และ update ตำแหน่งของ monster ใน monstersMap
- void tickTower (double dt)	Tick ทุก tower ในเกม
+ void handleClick (MouseEvent mouse Event)	<p>จัดการเวลาผู้ใช้คลิกบน canvas</p> <ol style="list-style-type: none"> 1. หากตอนนี้ผู้เล่นมีเลือก tower button จาก TowerSelectUI และกดตรงจุดบน tile ที่สร้าง tower ได้และมีเงินพอ จะสร้าง tower ขึ้นบน tile นั้น, หักเงิน, refresh sidebar, และเล่นเสียงตอนสร้าง tower 2. หากคลิกตรงที่มี tower อยู่จะทำการเลือก tower นั้นผ่าน Towers::select หากเลือกสำเร็จจะแสดง tower information นั้น
+ void addMonster (Monster monster)	เพิ่ม monster ลงในเกม

+ void addBullet (Bullets bullet)	เพิ่ม bullet ลงในเกม
+void addDmgIng (Dmg dmgInd)	เพิ่ม damage Indicator ลงในเกม
+ Set<Monstet> getMonsterAt (int x , int y)	หา monster ที่อยู่บน tile ณ ตำแหน่ง x,y
+ Set<Monster> getMonsterAt (Pair<Integer,Integer> pos)	หา monster ที่อยู่บน tile ณ ตำแหน่ง pos
+ getter each field as needed	

10.3 Class Main extends Application

10.3.1 Field

+ <u>Game game</u>	Current Game
+ <u>Canvas canvas</u>	Canvas ที่ไว้แสดงผล game
+ <u>InputUtils inputUtils</u>	InputUtils singleton
+ <u>Sidebar sidebar</u>	Sidebar UI
+ <u>Stage stage</u>	เก็บ stage ที่ได้มาเป็น argument ใน start
+ <u>AnimationTimer gameLoop</u>	AnimationTimer ที่สั่งควบคุมเรียกใช้ Draw::draw Tick::tick ใน game กับ fpsCounter ทำงานประมาณ 60 ครั้งใน 1 วินาที

10.3.2 Constructor

10.3.3 Method

+ void start (Stage primaryStage)	ตั้ง title เป็น Tower d' Fence ตั้ง root node ใน stage's scene เป็น new TitleScreen() แล้ว show stage
+ <u>void restart()</u>	Initialize canvas, sidebar, game, และ inputUtils เรียกใช้ setupUI, setupGraphics ตั้ง root node ใน stage's scene เป็น Hbox ที่มีลูกเป็น canvas

	และ sidebar
- <u>void setupUI ()</u>	ตั้งขนาดของ canvas เป็น 800,800 และเพิ่ม TowerButton ของ Type1, Type2, Type3 ลงไปใน sidebar.towerSelectUI
- <u>void setupGraphics</u> (GraphicsContext gc)	ตั้งให้ Fill เป็น Color.BLACK, ขนาด Font เป็น 18 ตั้งให้ mouse click บน canvas จะสั่งทำงาน game.handleClick สร้าง gameLoop และ fpsCounter
+ <u>void switchToEndScreen()</u>	หยุด gameLoop แล้ว ตั้ง root node ใน stage's scene เป็น new EndScreen()
+ <u>void main (String[] args)</u>	launch(args);

10.4 Interface Tick

10.4.1 Field

10.4.2 Constructor

10.4.3 Method

+ void tick (double dt)	การประมวลผลใน 1 frame เวลาระหว่าง frame นี้กับก่อนหน้าเป็น dt (หน่วยวินาที)
---------------------------	---