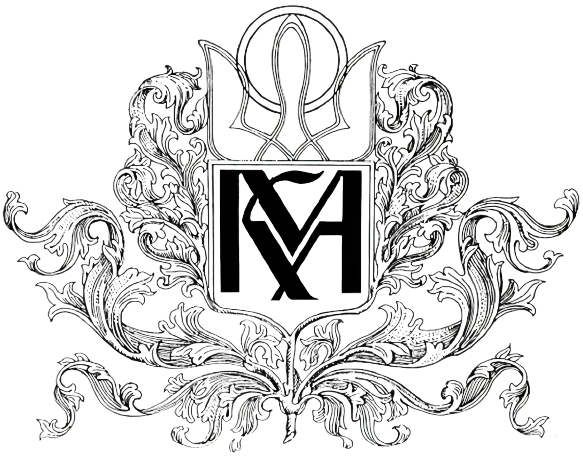
Національний університет "Києво-Могилянська Академія"
Факультет інформатики



Freshmaze

Виконала Команда "DAR": Омельчук Руслан, Сукайло Дмитро, Трохимчук Артем Студенти І курсу НаУКМА Факультет Інформатики Спеціальність: інженерія програмного забезпечення, гр.3
Викладач: Кирієнко Оксана Валентинівна

Київ — 2023

1. Постановка і аналіз задачі

Розробити гру, в процесі використовувати Git для розподіленої розробки програмного забезпечення. Реалізувати програму інтерактивної графічної гри, графічного інтерфейсу користувача.

1.1. Опис ігрового процесу

Freshmaze - це гра в жанрі roguelike, суть якої полягає у проходженні гравцем процедурно згенерованих рівнів підземелля, на якому він зустрічає ворогів та нагороди. У одній із кімнат розміщується перехід на наступний рівень, де збільшується складність. Гра може тривати або нескінченно (до смерті гравця) або мати обмежену кількість рівнів (це залежить від подальшої реалізації). Після смерті гравця ігровий процес починається спочатку.

1.2. Аналіз задачі

Особливістю створення ігор є динамічність розробки та широкий простір для змін, оскільки у процесі розробки може виявитися, що певні, коректно реалізовані, з погляду коду, механіки не є "цікавими" для гри або потребують значних змін. Тому на початковому етапі можна виділити лише основні підзадачі:

1. Процедурна генерація рівнів
2. Гравець та його переміщення
3. Система бою та штучний інтелект ворогів
4. Система нагород та бонусів

Не варто також забувати про створення графічних ресурсів та цілісного ігрового дизайну.

2. Розподіл ролей в середині команди

Згідно з названими вище причинами, тобто оскільки кожен аспект гри може зайняти непрогнозовану кількість часу, наша команда вирішила застосувати динамічний підхід у розподілі ролей. Проте є початкові напрямки роботи які обрали члени команди:

- Руслан був тімлідом, зайнявся генерацією ідей, та основними розробками гри;
- Артем — створив гравця та системи бою, налагодження переходу між рівнями;
- Дмитро — створив систему нагород, відеорекламу та звіт.

3. Структура програми

4. Структура програми

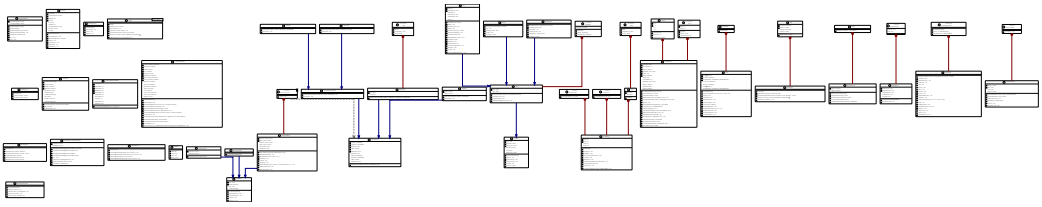


Рис. 1: Діаграма класів

5. Опис методів та класів

Freshmaze

Generated by Doxygen 1.9.6

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

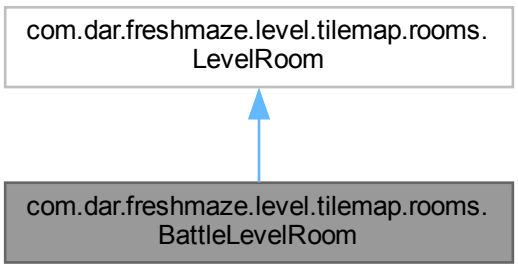
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

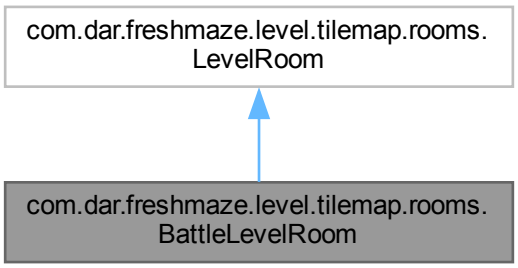
Class Documentation

3.1 com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom:



Collaboration diagram for com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom:



Classes

- interface **DynamicTileAction**

Public Member Functions

- **BattleLevelRoom** (Rectangle bounds, EnemyGenerator enemyGenerator, float spikeInterval)
- Array< Vector2 > **getEntrances** ()
- void addEntrance (Vector2 entrance)
- Array< Vector2 > **getSpikes** ()
- void **setSpikes** (Array< Vector2 > newSpikes)
- void act (float dt)
- void onDestroy ()
- void onPlayerEnter (Bob bob)
- void **onEnemyDeath** (Enemy enemy)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.rooms.LevelRoom

- **LevelRoom** (Rectangle bounds)
- Level **getLevel** ()
- void **setLevel** (Level newLevel)
- Rectangle **getBounds** ()
- void **act** (float dt)
- void **onDestroy** ()
- void **onPlayerEnter** (Bob bob)
- void **onPlayerExit** (Bob bob)

3.1.1 Detailed Description

Battle room class

3.1.2 Member Function Documentation

3.1.2.1 act()

```
void com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom.act (
            float dt )  [inline]
```

Reimplemented from com.dar.freshmaze.level.tilemap.rooms.LevelRoom.

3.1.2.2 addEntrance()

```
void com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom.addEntrance (
            Vector2 entrance )  [inline]
```

Add entrance to the room

Parameters

entrance	

3.1.2.3 onDestroy()

```
void com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom.onDestroy ( )  [inline]
```

Reimplemented from com.dar.freshmaze.level.tilemap.rooms.LevelRoom.

3.1.2.4 onPlayerEnter()

```
void com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom.onPlayerEnter (
            Bob bob )  [inline]
```

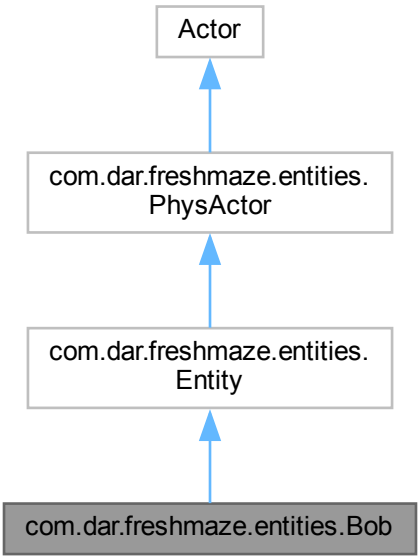
Reimplemented from com.dar.freshmaze.level.tilemap.rooms.LevelRoom.

The documentation for this class was generated from the following file:

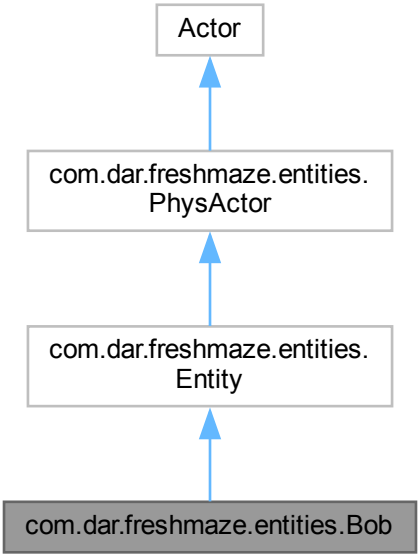
- core/src/com/dar/freshmaze/level/tilemap/rooms/BattleLevelRoom.java

3.2 com.dar.freshmaze.entities.Bob Class Reference

Inheritance diagram for com.dar.freshmaze.entities.Bob:



Collaboration diagram for com.dar.freshmaze.entities.Bob:



Public Member Functions

- **Bob** (World physWorld, Level level, Vector2 spawnPos)
- float getAttackTimeLeft ()
- float **getTimePerAttack** ()
- void damage (int damage)
- void heal (int amount)
- void increaseAttackSpeed (float amount)
- void draw (Batch batch, float alpha)
- void act (float delta)
- void **addObjectInRadius** (Object userData)
- void **removeObjectInRadius** (Object userData)
- void **setHealth** (int health)
- int **getHealth** ()
- int **getMaxHealth** ()

Public Member Functions inherited from com.dar.freshmaze.entities.Entity

- **Entity** (World physWorld, Sprite sprite, Body body, Vector2 spriteOffset, SpriteKind spriteKind, Vector2 spawnPos)
- Sprite **getSprite** ()
- void **draw** (Batch batch, float alpha)

Public Member Functions inherited from com.dar.freshmaze.entities.PhysActor

- **PhysActor** (World physWorld, Body body)
- Body **getBody** ()
- World **getPhysWorld** ()
- boolean **isDestroyed** ()
- void **teleport** (Vector2 pos)
- void **destroy** ()
- boolean **remove** ()
- void **act** (float delta)

Public Attributes

- boolean **movingRight** = false
- boolean **movingLeft** = false
- boolean **movingUp** = false
- boolean **movingDown** = false

Static Public Attributes

- static final float **MOVEMENT_SPEED** = 4.0f
- static final float **deltaPx** = 1.0f
- static final float **deltaPy** = 1.0f

Additional Inherited Members

Protected Member Functions inherited from com.dar.freshmaze.entities.Entity

- void **positionChanged** ()
- void **setShaderSortHeight** (Batch batch, float offset)

3.2.1 Detailed Description

Class that represents Bob The Player.

3.2.2 Member Function Documentation

3.2.2.1 act()

```
void com.dar.freshmaze.entities.Bob.act (
            float delta )  [inline]
```

Reimplemented from com.dar.freshmaze.entities.PhysActor.

3.2.2.2 damage()

```
void com.dar.freshmaze.entities.Bob.damage (
            int damage )  [inline]
```

Damage the bob

Parameters

damage	the delta for the health.

3.2.2.3 draw()

```
void com.dar.freshmaze.entities.Bob.draw (
            Batch batch,
            float alpha )  [inline]
```

Reimplemented from com.dar.freshmaze.entities.Entity.

3.2.2.4 getAttackTimeLeft()

```
float com.dar.freshmaze.entities.Bob.getAttackTimeLeft ( )  [inline]
```

Returns the attack time left

Returns

attack time left

3.2.2.5 heal()

```
void com.dar.freshmaze.entities.Bob.heal (
            int amount )  [inline]
```

Heal the bob

Parameters

amount	the delta for the heal

3.2.2.6 increaseAttackSpeed()

```
void com.dar.freshmaze.entities.Bob.increaseAttackSpeed (
            float amount )  [inline]
```

Increase the attack speed

Parameters

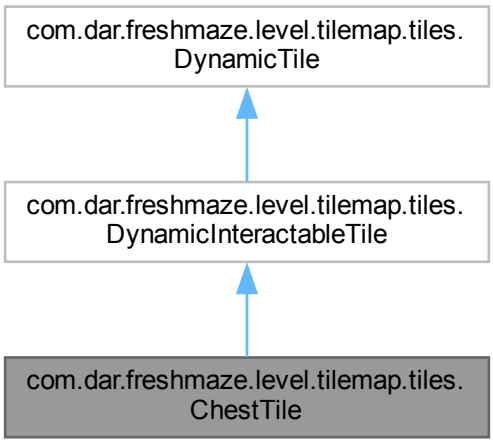
amount	the amount to increase the attack speed

The documentation for this class was generated from the following file:

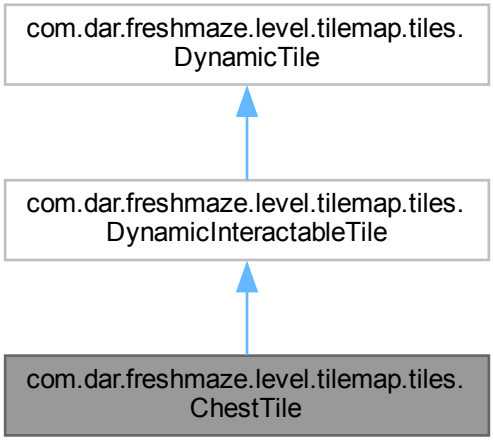
- core/src/com/dar/freshmaze/entities/Bob.java

3.3 com.dar.freshmaze.level.tilemap.tiles.ChestTile Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.tiles.ChestTile:



Collaboration diagram for com.dar.freshmaze.level.tilemap.tiles.ChestTile:



Public Member Functions

- **ChestTile** (LevelTilemap tilemap, LevelTilemap.CellPos pos, TiledMapTile closedTile, TiledMapTile open↩ Tile)
- void interact (Bob player)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicInteractableTile

- **DynamicInteractableTile** (LevelTilemap tilemap, LevelTilemap.CellPos pos, TiledMapTile defaultTile, LevelTilemap.Layer defaultLayer)
- abstract void **interact** (Bob player)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

- **DynamicTile** (LevelTilemap tilemap, LevelTilemap.CellPos cellPos, TiledMapTile defaultTile, Level↩ Tilemap.Layer defaultLayer)
- LevelTilemap **getTilemap** ()
- Body **getPhysBody** ()
- LevelTilemap.CellPos **getCellPos** ()
- TiledMapTile **getDefaultTile** ()
- LevelTilemap.Layer **getDefaultLayer** ()

Additional Inherited Members

Protected Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

- void **setPhysBody** (Body newPhysBody)

3.3.1 Member Function Documentation

3.3.1.1 interact()

```
void com.dar.freshmaze.level.tilemap.tiles.ChestTile.interact (
            Bob player )  [inline]
```

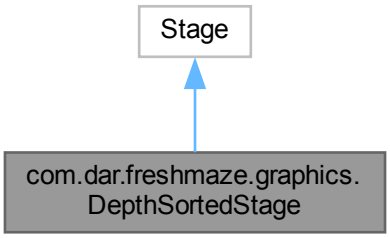
Reimplemented from com.dar.freshmaze.level.tilemap.tiles.DynamicInteractableTile.

The documentation for this class was generated from the following file:

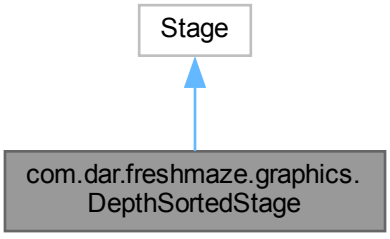
- core/src/com/dar/freshmaze/level/tilemap/tiles/ChestTile.java

3.4 com.dar.freshmaze.graphics.DepthSortedStage Class Reference

Inheritance diagram for com.dar.freshmaze.graphics.DepthSortedStage:



Collaboration diagram for com.dar.freshmaze.graphics.DepthSortedStage:



Public Member Functions

- DepthSortedStage (Viewport viewport)
- DepthSortedStage (Viewport viewport, Batch batch)
- Vector2 **getVerticalViewBounds** ()
- void **setVerticalViewBounds** (Vector2 newVerticalViewBounds)
- void **shaderSetVerticalViewBounds** ()
- void **draw** ()

3.4.1 Detailed Description

Modified stage, that renders object using OpenGL depth test

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DepthSortedStage() [1/2]

```
com.dar.freshmaze.graphics.DepthSortedStage.DepthSortedStage (
            Viewport viewport )  [inline]
```

Creates a stage with the specified viewport. The stage will use its own `Batch` which will be disposed when the stage is disposed.

3.4.2.2 DepthSortedStage() [2/2]

```
com.dar.freshmaze.graphics.DepthSortedStage.DepthSortedStage (
            Viewport viewport,
            Batch batch )  [inline]
```

Creates a stage with the specified viewport and batch. This can be used to specify an existing batch or to customize which batch implementation is used.

Parameters

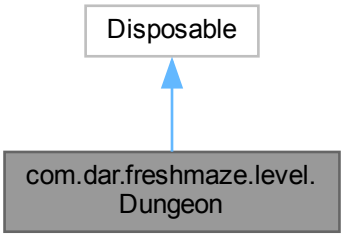
batch	Will not be disposed if `dispose()` is called, handle disposal yourself.

The documentation for this class was generated from the following file:

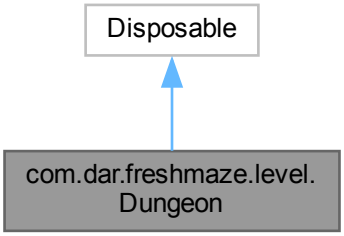
- core/src/com/dar/freshmaze/graphics/DepthSortedStage.java

3.5 com.dar.freshmaze.level.Dungeon Class Reference

Inheritance diagram for com.dar.freshmaze.level.Dungeon:



Collaboration diagram for com.dar.freshmaze.level.Dungeon:



Public Member Functions

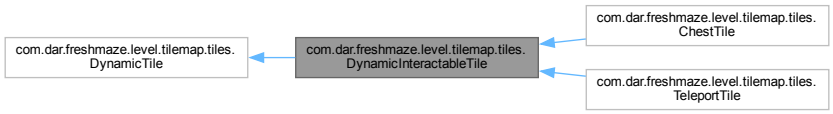
- **Dungeon** (Level level, Bob bob)
- Level **getLevel** ()
- Bob **getBob** ()
- boolean **isPendingTransition** ()
- void **moveToNextLevel** ()
- void **update** (float dt)
- boolean **isMaxLevel** ()
- int **getLevelIndex** ()
- void **dispose** ()

The documentation for this class was generated from the following file:

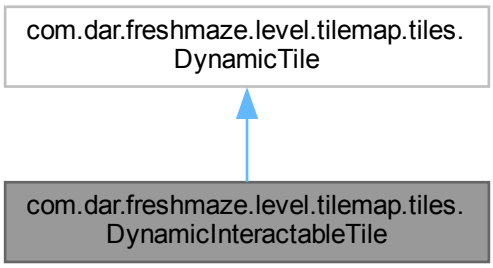
- core/src/com/dar/freshmaze/level/Dungeon.java

3.6 com.dar.freshmaze.level.tilemap.tiles.DynamicInteractableTile Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.tiles.DynamicInteractableTile:



Collaboration diagram for com.dar.freshmaze.level.tilemap.tiles.DynamicInteractableTile:



Public Member Functions

- **DynamicInteractableTile** (LevelTilemap tilemap, LevelTilemap.CellPos pos, TiledMapTile defaultTile, LevelTilemap.Layer defaultLayer)
- abstract void **interact** (Bob player)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

- **DynamicTile** (LevelTilemap tilemap, LevelTilemap.CellPos cellPos, TiledMapTile defaultTile, Level↩ Tilemap.Layer defaultLayer)
- LevelTilemap **getTilemap** ()
- Body **getPhysBody** ()
- LevelTilemap.CellPos **getCellPos** ()
- TiledMapTile **getDefaultTile** ()
- LevelTilemap.Layer **getDefaultLayer** ()

Additional Inherited Members

Protected Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

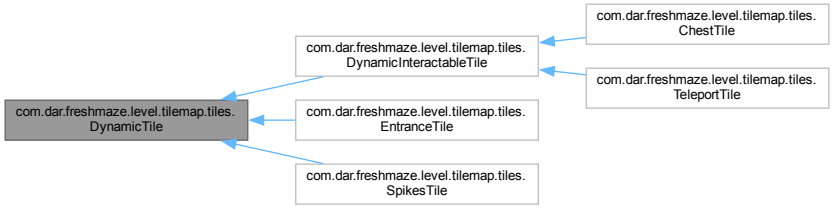
- void **setPhysBody** (Body newPhysBody)

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/tilemap/tiles/DynamicInteractableTile.java

3.7 com.dar.freshmaze.level.tilemap.tiles.DynamicTile Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.tiles.DynamicTile:



Public Member Functions

- **DynamicTile** (LevelTilemap tilemap, LevelTilemap.CellPos cellPos, TiledMapTile defaultTile, LevelTilemap.Layer defaultLayer)
- LevelTilemap **getTilemap** ()
- Body **getPhysBody** ()
- LevelTilemap.CellPos **getCellPos** ()
- TiledMapTile **getDefaultTile** ()
- LevelTilemap.Layer **getDefaultLayer** ()

Protected Member Functions

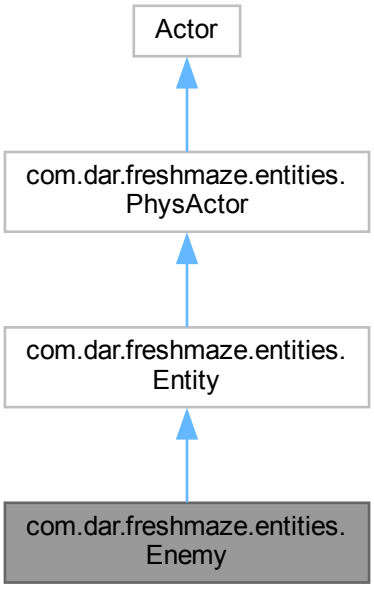
- void **setPhysBody** (Body newPhysBody)

The documentation for this class was generated from the following file:

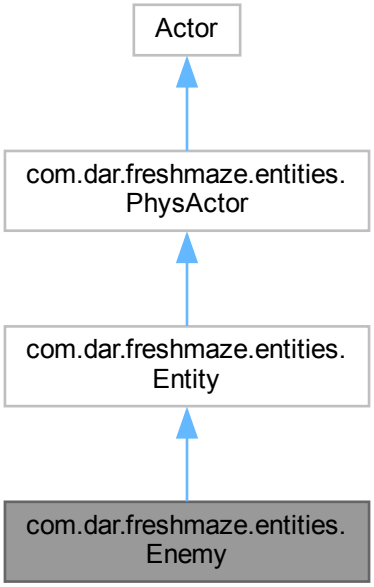
- core/src/com/dar/freshmaze/level/tilemap/tiles/DynamicTile.java

3.8 com.dar.freshmaze.entities.Enemy Class Reference

Inheritance diagram for com.dar.freshmaze.entities.Enemy:



Collaboration diagram for com.dar.freshmaze.entities.Enemy:



Public Member Functions

- **Enemy** (World physWorld, BattleLevelRoom room, Vector2 spawnPos)
- void **kill** ()
- void act (float delta)

Public Member Functions inherited from com.dar.freshmaze.entities.Entity

- **Entity** (World physWorld, Sprite sprite, Body body, Vector2 spriteOffset, SpriteKind spriteKind, Vector2 spawnPos)
- Sprite **getSprite** ()
- void **draw** (Batch batch, float alpha)

Public Member Functions inherited from com.dar.freshmaze.entities.PhysActor

- **PhysActor** (World physWorld, Body body)
- Body **getBody** ()
- World **getPhysWorld** ()
- boolean **isDestroyed** ()
- void **teleport** (Vector2 pos)
- void **destroy** ()
- boolean **remove** ()
- void **act** (float delta)

Public Attributes

- final float **movementSpeed**
- final float **deltaPx**
- final float **deltaPy**

Additional Inherited Members

Protected Member Functions inherited from com.dar.freshmaze.entities.Entity

- void **positionChanged** ()
- void **setShaderSortHeight** (Batch batch, float offset)

3.8.1 Member Function Documentation

3.8.1.1 act()

```
void com.dar.freshmaze.entities.Enemy.act (
            float delta )  [inline]
```

Reimplemented from com.dar.freshmaze.entities.PhysActor.

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/entities/Enemy.java

3.9 com.dar.freshmaze.level.EnemyGenerator Class Reference

Classes

- class **Result**

Public Member Functions

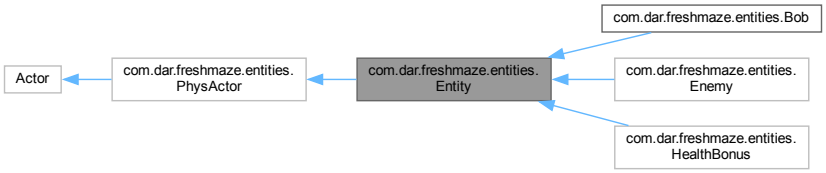
- **EnemyGenerator** (World physWorld, Stage stage)
- void **setDungeon** (Dungeon dungeon)
- Result **generate** (BattleLevelRoom room)

The documentation for this class was generated from the following file:

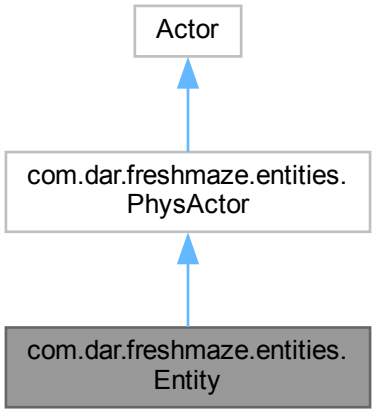
- core/src/com/dar/freshmaze/level/EnemyGenerator.java

3.10 com.dar.freshmaze.entities.Entity Class Reference

Inheritance diagram for com.dar.freshmaze.entities.Entity:



Collaboration diagram for com.dar.freshmaze.entities.Entity:



Classes

- enum SpriteKind

Public Member Functions

- **Entity** (World physWorld, Sprite sprite, Body body, Vector2 spriteOffset, SpriteKind spriteKind, Vector2 spawnPos)
- Sprite **getSprite** ()
- void **draw** (Batch batch, float alpha)

Public Member Functions inherited from com.dar.freshmaze.entities.PhysActor

- **PhysActor** (World physWorld, Body body)
- Body **getBody** ()
- World **getPhysWorld** ()
- boolean **isDestroyed** ()
- void **teleport** (Vector2 pos)
- void **destroy** ()
- boolean **remove** ()
- void **act** (float delta)

Protected Member Functions

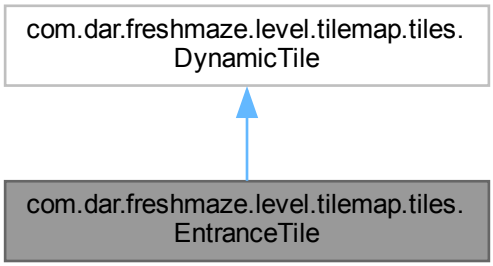
- void **positionChanged** ()
- void **setShaderSortHeight** (Batch batch, float offset)

The documentation for this class was generated from the following file:

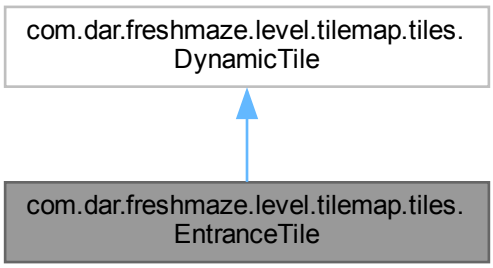
- core/src/com/dar/freshmaze/entities/Entity.java

3.11 com.dar.freshmaze.level.tilemap.tiles.EntranceTile Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.tiles.EntranceTile:



Collaboration diagram for com.dar.freshmaze.level.tilemap.tiles.EntranceTile:



Classes

- enum State

Public Member Functions

- **EntranceTile** (LevelTilemap tilemap, LevelTilemap.CellPos pos, TiledMapTile openTile, TiledMapTile closedTile, TiledMapTile clearedTile)
- State **getState** ()
- void **setState** (State newState)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

- **DynamicTile** (LevelTilemap tilemap, LevelTilemap.CellPos cellPos, TiledMapTile defaultTile, Level↩ Tilemap.Layer defaultLayer)
- LevelTilemap **getTilemap** ()
- Body **getPhysBody** ()
- LevelTilemap.CellPos **getCellPos** ()
- TiledMapTile **getDefaultTile** ()
- LevelTilemap.Layer **getDefaultLayer** ()

Additional Inherited Members

Protected Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

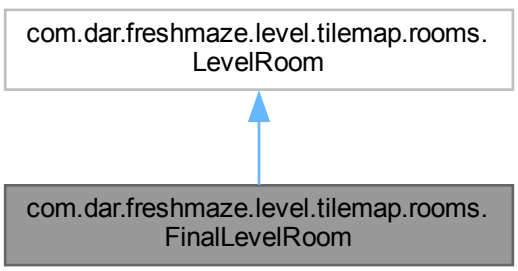
- void **setPhysBody** (Body newPhysBody)

The documentation for this class was generated from the following file:

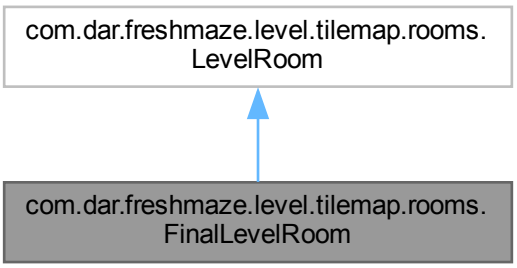
- core/src/com/dar/freshmaze/level/tilemap/tiles/EntranceTile.java

3.12 com.dar.freshmaze.level.tilemap.rooms.FinalLevelRoom Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.rooms.FinalLevelRoom:



Collaboration diagram for com.dar.freshmaze.level.tilemap.rooms.FinalLevelRoom:



Public Member Functions

- **FinalLevelRoom** (Rectangle bounds, Vector2 teleportPos)
- final Vector2 **getTeleportPos** ()

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.rooms.LevelRoom

- **LevelRoom** (Rectangle bounds)
- Level **getLevel** ()
- void **setLevel** (Level newLevel)
- Rectangle **getBounds** ()
- void **act** (float dt)
- void **onDestroy** ()
- void **onPlayerEnter** (Bob bob)
- void **onPlayerExit** (Bob bob)

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/tilemap/rooms/FinalLevelRoom.java

3.13 com.dar.freshmaze.indicator.RectIndicator.FloatRangeBinder Interface Reference

Public Member Functions

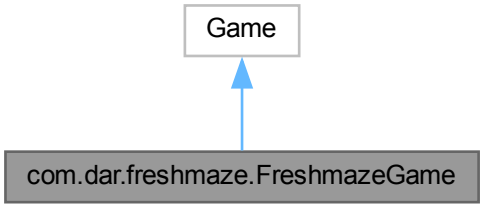
- float **getValue** ()
- float **getMaxValue** ()

The documentation for this interface was generated from the following file:

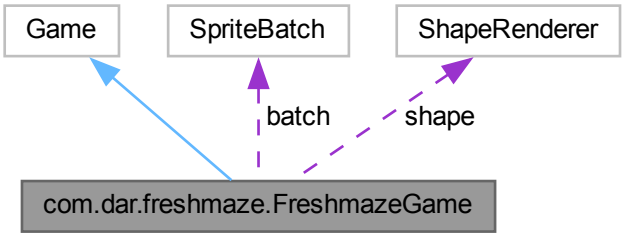
- core/src/com/dar/freshmaze/indicator/RectIndicator.java

3.14 com.dar.freshmaze.FreshmazeGame Class Reference

Inheritance diagram for com.dar.freshmaze.FreshmazeGame:



Collaboration diagram for com.dar.freshmaze.FreshmazeGame:



Public Member Functions

- void **create** ()
- void **start** (boolean isRestart)
- void **dispose** ()

Public Attributes

- SpriteBatch **batch**
- ShapeRenderer **shape**

Static Public Attributes

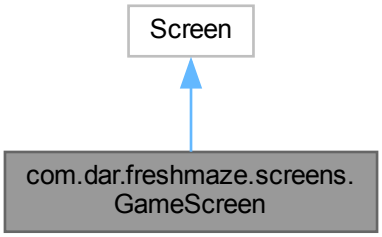
- static final float **WIDTH** = 1280
- static final float **HEIGHT** = 720

The documentation for this class was generated from the following file:

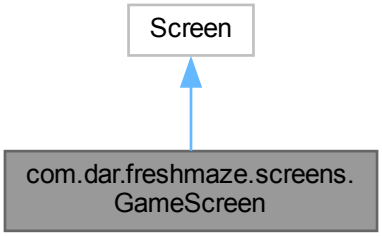
- core/src/com/dar/freshmaze/FreshmazeGame.java

3.15 com.dar.freshmaze.screens.GameScreen Class Reference

Inheritance diagram for com.dar.freshmaze.screens.GameScreen:



Collaboration diagram for com.dar.freshmaze.screens.GameScreen:



Public Member Functions

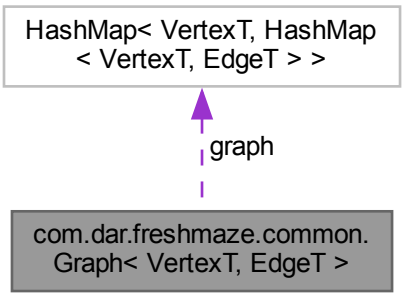
- **GameScreen** (FreshmazeGame game, OrthographicCamera camera, Viewport viewport, boolean skip↩ Menu)
- void **show** ()
- void **render** (float delta)
- void **resize** (int width, int height)
- void **pause** ()
- void **resume** ()
- void **hide** ()
- void **dispose** ()

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/screens/GameScreen.java

3.16 com.dar.freshmaze.common.Graph< VertexT, EdgeT > Class Template Reference

Collaboration diagram for com.dar.freshmaze.common.Graph< VertexT, EdgeT >:



Public Member Functions

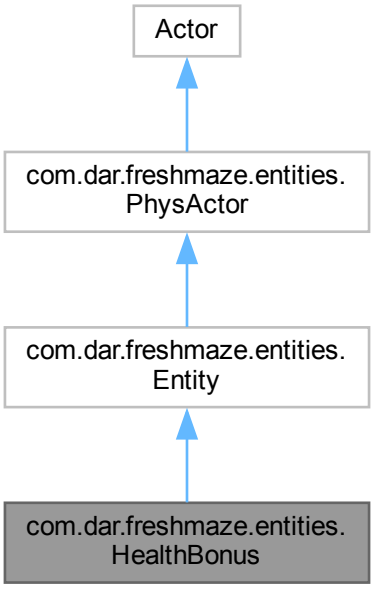
- Set< Map.Entry< VertexT, HashMap< VertexT, EdgeT > > > **entrySet** ()
- Map< VertexT, EdgeT > **getConnections** (VertexT vertex)
- boolean **contains** (VertexT first)
- void **add** (VertexT first, VertexT second, EdgeT edge)
- void **addDirected** (VertexT first, VertexT second, EdgeT edge)
- void **remove** (VertexT vertex)

The documentation for this class was generated from the following file:

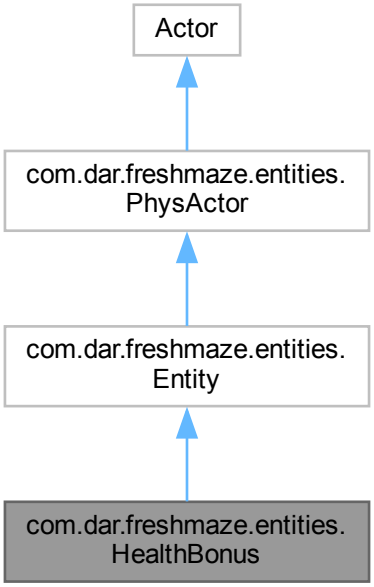
- core/src/com/dar/freshmaze/common/Graph.java

3.17 com.dar.freshmaze.entities.HealthBonus Class Reference

Inheritance diagram for com.dar.freshmaze.entities.HealthBonus:



Collaboration diagram for com.dar.freshmaze.entities.HealthBonus:



Public Member Functions

- **HealthBonus** (World physWorld, BattleLevelRoom room, Vector2 spawnPos)
- void **interact** (Bob bob)
- void draw (Batch batch, float alpha)

Public Member Functions inherited from com.dar.freshmaze.entities.Entity

- **Entity** (World physWorld, Sprite sprite, Body body, Vector2 spriteOffset, SpriteKind spriteKind, Vector2 spawnPos)
- Sprite **getSprite** ()
- void **draw** (Batch batch, float alpha)

Public Member Functions inherited from com.dar.freshmaze.entities.PhysActor

- **PhysActor** (World physWorld, Body body)
- Body **getBody** ()
- World **getPhysWorld** ()
- boolean **isDestroyed** ()
- void **teleport** (Vector2 pos)
- void **destroy** ()
- boolean **remove** ()
- void **act** (float delta)

Additional Inherited Members

Protected Member Functions inherited from com.dar.freshmaze.entities.Entity

- void **positionChanged** ()
- void **setShaderSortHeight** (Batch batch, float offset)

3.17.1 Member Function Documentation

3.17.1.1 draw()

```
void com.dar.freshmaze.entities.HealthBonus.draw (
            Batch batch,
            float alpha )  [inline]
```

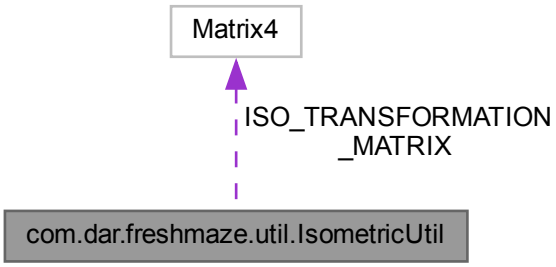
Reimplemented from com.dar.freshmaze.entities.Entity.

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/entities/HealthBonus.java

3.18 com.dar.freshmaze.util.IsometricUtil Class Reference

Collaboration diagram for com.dar.freshmaze.util.IsometricUtil:



Static Public Member Functions

- static Vector2 isoToCart (Vector2 vec)
- static Vector2 cartToIso (Vector2 vec)

Static Public Attributes

- static final Matrix4 ISO_TRANSFORMATION_MATRIX

3.18.1 Detailed Description

Class with various helper methods for woring with isometric coordinate system

3.18.2 Member Function Documentation

3.18.2.1 cartToIso()

```
static Vector2 com.dar.freshmaze.util.IsometricUtil.cartToIso (
            Vector2 vec )  [inline], [static]
```

Converts vector from cartesian to isometric coordinate system.

The original formula is the following:

```
  new Vector2(
vec.x - vec.y,
0.5f * (vec.x + vec.y)
);
```

But we need to rotate it to match our tilemap. The result is:

```
return new Vector2(
vec.x + vec.y,
0.5f * (vec.x - vec.y)
);
```

NOTE: I have no idea where multiplication of both x and y by 0.5 comes from, it makes no sense. Fix it if possible

Parameters

vec	vector in cartesian system

Returns

vector in isometric system

3.18.2.2 isoToCart()

```
static Vector2 com.dar.freshmaze.util.IsometricUtil.isoToCart (
            Vector2 vec )  [inline], [static]
```

Converts vector from isometric to cartesian coordinate system.

The original formula is the following:

```
  new Vector2(
0.5f * (2.0f * vec.y + vec.x),
0.5f * (2.0f * vec.y - vec.x)
);
```

But we need to rotate it to match our tilemap. The result is:

```
return new Vector2(
0.5f * (-2.0f * vec.y + vec.x),
0.5f * (2.0f * vec.y + vec.x)
);
```

Parameters

vec	vector in isometric system

Returns

vector in cartesian system

3.18.3 Member Data Documentation

3.18.3.1 ISO_TRANSFORMATION_MATRIX

```
final Matrix4 com.dar.freshmaze.util.IsometricUtil.ISO_TRANSFORMATION_MATRIX  [static]
```

Matrix that can be used to transform coordinates to isometric for rendering (ot other purposes). Doesn't need to be used if the sprites are already drawn as isometric.

Can be used like:

```
game.batch.setTransformMatrix(IsometricUtil.ISO_TRANSFORMATION_MATRIX);
```

or

```
physDebugRenderer.render(physWorld, camera.combined.mul(IsometricUtil.ISO_TRANSFORMATION_MATRIX))
```

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/util/IsometricUtil.java

3.19 com.dar.freshmaze.level.bitmap.LevelBitmap.Cell.Kind Enum Reference

Public Attributes

- **Empty**
- **Room**
- **Hall**
- **Wall**
- **HallEntrance**
- **Teleport**
- **Spikes**

The documentation for this enum was generated from the following file:

- core/src/com/dar/freshmaze/level/bitmap/LevelBitmap.java

3.20 com.dar.freshmaze.level.tilemap.LevelTilemap.Layer Enum Reference

Public Member Functions

- int **getIndex** ()
- **Layer** (int index)

Public Attributes

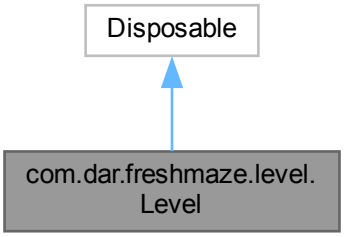
- **Floor** =(0)
- **FloorOverlay** =(1)
- **Wall** =(2)

The documentation for this enum was generated from the following file:

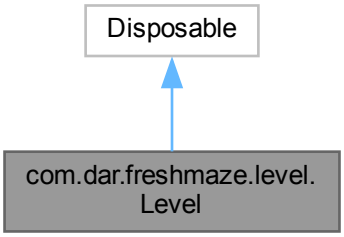
- core/src/com/dar/freshmaze/level/tilemap/LevelTilemap.java

3.21 com.dar.freshmaze.level.Level Class Reference

Inheritance diagram for com.dar.freshmaze.level.Level:



Collaboration diagram for com.dar.freshmaze.level.Level:



Classes

- class **DebugRender**

Public Member Functions

- **Level** (World physWorld, Stage stage)
- LevelRoom **generate** (int levelIndex)
- int **getMaxLevel** ()
- List< LevelRoom > **getRooms** ()
- LevelTilemap **getTilemap** ()
- SortedIsometricTiledMapRenderer **getTilemapRenderer** ()
- void **update** (float dt)
- void **render** (OrthographicCamera camera, float dt, int[ ] layers, boolean writeDepth)
- void **debugRender** (Camera camera, float dt, int kind)
- void **dispose** ()

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/Level.java

3.22 com.dar.freshmaze.level.bitmap.LevelBitmap Class Reference

Classes

- class **Cell**

Public Member Functions

- int **getWidth** ()
- int **getHeight** ()
- void **generate** (LevelNodeGenerator generator)
- Cell **getCell** (int xi, int yi)
- Cell **getCell** (int index)

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/bitmap/LevelBitmap.java

3.23 com.dar.freshmaze.level.graph.LevelGraph Class Reference

Classes

- class **Edge**

Public Member Functions

- Set< Map.Entry< LevelNode, HashMap< LevelNode, Edge > > > **entrySet** ()
- void generate (List< LevelNode > leaves)

3.23.1 Detailed Description

Class that represents the graph of the levels.

3.23.2 Member Function Documentation

3.23.2.1 generate()

```
void com.dar.freshmaze.level.graph.LevelGraph.generate (
            List< LevelNode > leaves ) [inline]
```

Generate level graph from leaves (list of level nodes)

Parameters

leaves	the leaves

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/graph/LevelGraph.java

3.24 com.dar.freshmaze.level.graph.LevelNode Class Reference

Public Member Functions

- **LevelNode** (Rectangle bounds, LevelNodeGenerationRules rules)
- boolean **isLeaf** ()
- Rectangle **getBounds** ()
- LevelNode **getLeftChild** ()
- LevelNode **getRightChild** ()
- Rectangle **getRoomBounds** ()
- boolean **split** ()
- void **generateRoom** ()

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/graph/LevelNode.java

3.25 com.dar.freshmaze.level.graph.LevelNodeGenerationRules Class Reference

Public Member Functions

- **LevelNodeGenerationRules** (int minRoomSize, int minNodeSize, int maxNodeSize, float splitChance, int roomGap)
- int **getMinRoomSize** ()
- int **getMinNodeSize** ()
- int **getMaxNodeSize** ()
- float **getSplitChance** ()
- int **getRoomGap** ()

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/graph/LevelNodeGenerationRules.java

3.26 com.dar.freshmaze.level.graph.LevelNodeGenerator Class Reference

Public Member Functions

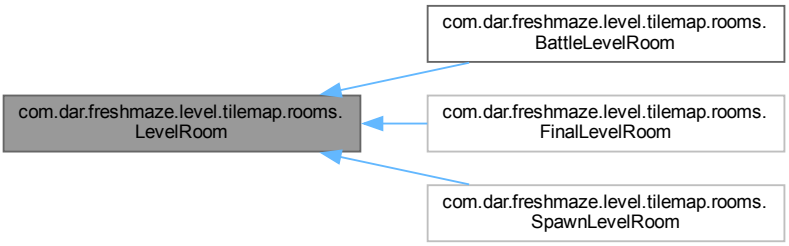
- void **generate** (Vector2 levelSize, int hallThickness, LevelNodeGenerationRules rules, EnemyGenerator enemyGenerator, SpikeGenerator spikeGenerator)
- Vector2 **getLevelSize** ()
- LevelNode **getRoot** ()
- List< LevelNode > **getLeaves** ()
- List< LevelRoom > **getRooms** ()
- LevelGraph **getGraph** ()
- List< Rectangle > **getHalls** ()
- LevelRoom **getSpawnRoom** ()
- LevelRoom **getFinalRoom** ()

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/graph/LevelNodeGenerator.java

3.27 com.dar.freshmaze.level.tilemap.rooms.LevelRoom Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.rooms.LevelRoom:



Public Member Functions

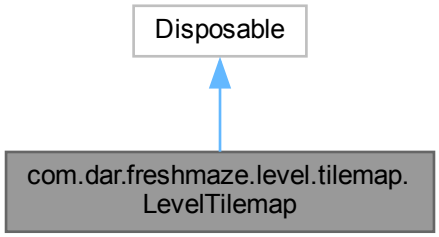
- **LevelRoom** (Rectangle bounds)
- Level **getLevel** ()
- void **setLevel** (Level newLevel)
- Rectangle **getBounds** ()
- void **act** (float dt)
- void **onDestroy** ()
- void **onPlayerEnter** (Bob bob)
- void **onPlayerExit** (Bob bob)

The documentation for this class was generated from the following file:

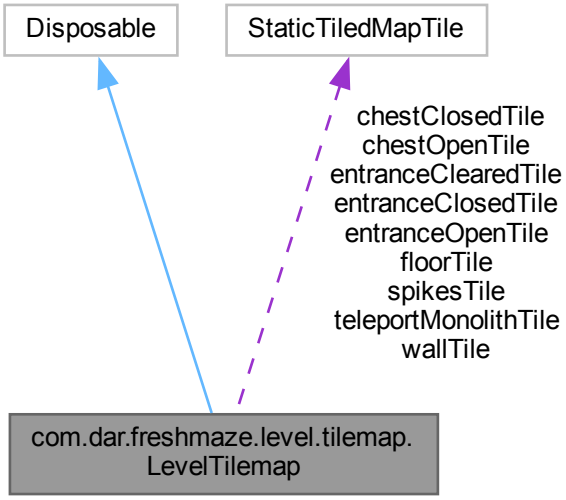
- core/src/com/dar/freshmaze/level/tilemap/rooms/LevelRoom.java

3.28 com.dar.freshmaze.level.tilemap.LevelTilemap Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.LevelTilemap:



Collaboration diagram for com.dar.freshmaze.level.tilemap.LevelTilemap:



Classes

- class **CellPos**
- enum Layer

Public Member Functions

- LevelTilemap (World physWorld, String tilesetPath, float tileSize, int textureTileSize)
- float **getTileSize** ()
- int **getTextureTileSize** ()
- TiledMap **getTilemap** ()
- Vector2 **cellPosToVec** (Vector2 cellPos)
- Vector2 **cellPosToVec** (CellPos cellPos)
- CellPos **vecToCellPos** (Vector2 pos)
- Vector2 **vecToCellPosVec** (Vector2 pos)
- World **getPhysWorld** ()
- Array< Body > **getPhysBodies** ()
- void **setDungeon** (Dungeon newDungeon)
- Dungeon **getDungeon** ()
- void **generate** (LevelBitmap bitmap)
- DynamicTile **getDynamicTile** (CellPos cellPos)
- void **placeDynamicTile** (DynamicTile dynamicTile)
- void **placeTile** (CellPos pos, TiledMapTile tile, Layer layerIndex)
- Body **createTilePhysBody** (CellPos pos, TiledMapTile tile)
- Body **createTilePhysBodySensor** (CellPos pos, TiledMapTile tile)
- void **dispose** ()

Public Attributes

- final StaticTiledMapTile **floorTile**
- final StaticTiledMapTile **wallTile**
- final StaticTiledMapTile **entranceOpenTile**
- final StaticTiledMapTile **entranceClearedTile**
- final StaticTiledMapTile **entranceClosedTile**
- final StaticTiledMapTile **teleportMonolithTile**
- final StaticTiledMapTile **chestClosedTile**
- final StaticTiledMapTile **chestOpenTile**
- final StaticTiledMapTile **spikesTile**

3.28.1 Detailed Description

TileMap for levels.

3.28.2 Constructor & Destructor Documentation

3.28.2.1 LevelTilemap()

```
com.dar.freshmaze.level.tilemap.LevelTilemap.LevelTilemap (
            World physWorld,
            String tilesetPath,
            float tileSize,
            int textureTileSize )  [inline]
```

Parameters

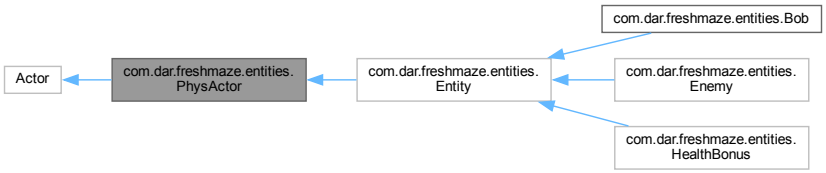
physWorld	
tilesetPath	
tileSize	
textureTileSize	

The documentation for this class was generated from the following file:

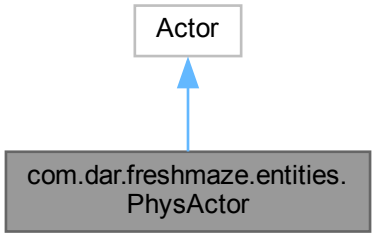
- core/src/com/dar/freshmaze/level/tilemap/LevelTilemap.java

3.29 com.dar.freshmaze.entities.PhysActor Class Reference

Inheritance diagram for com.dar.freshmaze.entities.PhysActor:



Collaboration diagram for com.dar.freshmaze.entities.PhysActor:



Public Member Functions

- **PhysActor** (World physWorld, Body body)
- Body **getBody** ()
- World **getPhysWorld** ()
- boolean **isDestroyed** ()
- void **teleport** (Vector2 pos)

- void **destroy** ()
- boolean **remove** ()
- void **act** (float delta)

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/entities/PhysActor.java

3.30 com.dar.freshmaze.util.RectangleUtil Class Reference

Static Public Member Functions

- static Vector2 **getRandomPoint** (Rectangle rect)
- static Rectangle expand (Rectangle rect, Vector2 delta)
- static Rectangle shrink (Rectangle rect, Vector2 delta)
- static Rectangle normalize (Rectangle rect)
- static boolean **containsExclusive** (Rectangle rect, Vector2 point)

3.30.1 Member Function Documentation

3.30.1.1 expand()

```
static Rectangle com.dar.freshmaze.util.RectangleUtil.expand (
            Rectangle rect,
            Vector2 delta )  [inline], [static]
```

Expands rectangle by delta in all directions

Parameters

rect	rectangle to expand
delta	size by which to expand the rectangle

Returns

rect

3.30.1.2 normalize()

```
static Rectangle com.dar.freshmaze.util.RectangleUtil.normalize (
            Rectangle rect )  [inline], [static]
```

Normalizes a rectangle, in other words, makes sure it has strictly positive size and modifies its position accordingly

Parameters

rect	rectangle to normalize

Returns

rect

3.30.1.3 shrink()

```
static Rectangle com.dar.freshmaze.util.RectangleUtil.shrink (
            Rectangle rect,
            Vector2 delta )  [inline], [static]
```

Shrinks rectangle by delta in all directions

Parameters

rect	rectangle to shrink
delta	size by which to shrink the rectangle

Returns

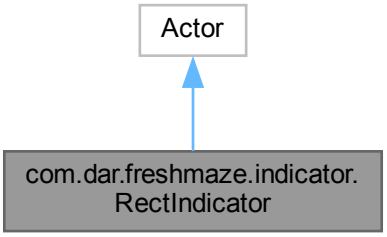
rect

The documentation for this class was generated from the following file:

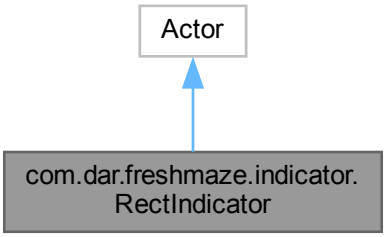
- core/src/com/dar/freshmaze/util/RectangleUtil.java

3.31 com.dar.freshmaze.indicator.RectIndicator Class Reference

Inheritance diagram for com.dar.freshmaze.indicator.RectIndicator:



Collaboration diagram for com.dar.freshmaze.indicator.RectIndicator:



Classes

- interface FloatRangeBinder

Public Member Functions

- **RectIndicator** (FloatRangeBinder valueBinder)
- void **setBackgroundColor** (Color newBackgroundColor)
- void **setIndicatorColor** (Color newIndicatorColor)
- void **draw** (Batch batch, float parentAlpha)

Protected Member Functions

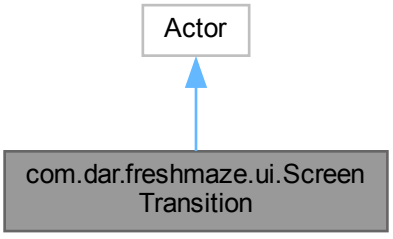
- float **getNormValue** ()

The documentation for this class was generated from the following file:

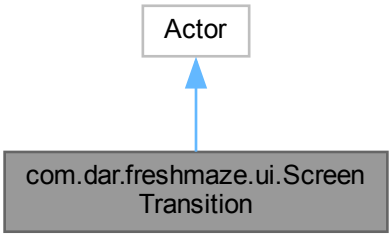
- core/src/com/dar/freshmaze/indicator/RectIndicator.java

3.32 com.dar.freshmaze.ui.ScreenTransition Class Reference

Inheritance diagram for com.dar.freshmaze.ui.ScreenTransition:



Collaboration diagram for com.dar.freshmaze.ui.ScreenTransition:



Classes

- interface TransitionCallback

Public Member Functions

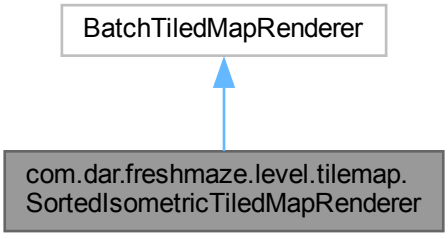
- **ScreenTransition** (float speed, float duration, boolean fadeOut)
- **ScreenTransition** (float speed, float duration, boolean fadeOut, TransitionCallback callback)
- void **setColor** (Color newColor)
- boolean **isFrozen** ()
- void **setIsFrozen** (boolean newIsFrozen)
- void **act** (float delta)
- void **draw** (Batch batch, float parentAlpha)

The documentation for this class was generated from the following file:

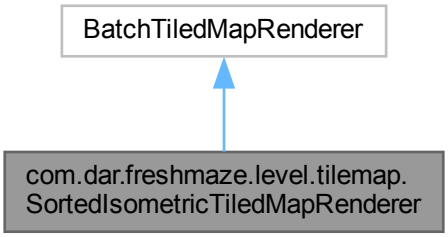
- core/src/com/dar/freshmaze/ui/ScreenTransition.java

3.33 com.dar.freshmaze.level.tilemap.SortedIsometricTiledMapRenderer Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.SortedIsometricTiledMapRenderer:



Collaboration diagram for com.dar.freshmaze.level.tilemap.SortedIsometricTiledMapRenderer:



Public Member Functions

- **SortedIsometricTiledMapRenderer** (TiledMap map)
- **SortedIsometricTiledMapRenderer** (TiledMap map, Batch batch)
- **SortedIsometricTiledMapRenderer** (TiledMap map, float unitScale)
- **SortedIsometricTiledMapRenderer** (TiledMap map, float unitScale, Batch batch)
- void **setWriteDepth** (boolean newWriteDepth)
- void **renderTileLayer** (TiledMapTileLayer layer)

3.33.1 Detailed Description

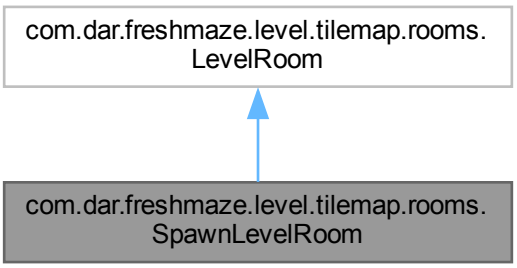
Isometric tilemap renderer that writes to the depth buffer if `writeDepth` flag is enabled

The documentation for this class was generated from the following file:

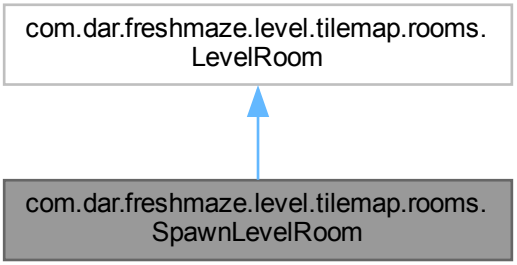
- core/src/com/dar/freshmaze/level/tilemap/SortedIsometricTiledMapRenderer.java

3.34 com.dar.freshmaze.level.tilemap.rooms.SpawnLevelRoom Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.rooms.SpawnLevelRoom:



Collaboration diagram for com.dar.freshmaze.level.tilemap.rooms.SpawnLevelRoom:



Public Member Functions

- **SpawnLevelRoom** (Rectangle bounds)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.rooms.LevelRoom

- **LevelRoom** (Rectangle bounds)
- Level **getLevel** ()
- void **setLevel** (Level newLevel)
- Rectangle **getBounds** ()
- void **act** (float dt)
- void **onDestroy** ()
- void **onPlayerEnter** (Bob bob)
- void **onPlayerExit** (Bob bob)

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/tilemap/rooms/SpawnLevelRoom.java

3.35 com.dar.freshmaze.level.tilemap.SpikeGenerator Class Reference

Public Member Functions

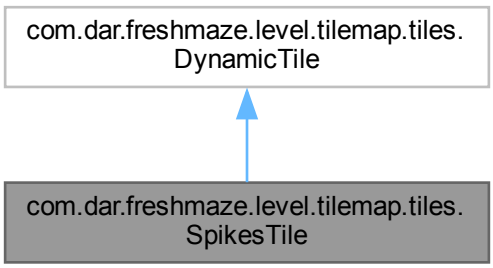
- Array< Vector2 > **generateSpikes** (LevelRoom room)

The documentation for this class was generated from the following file:

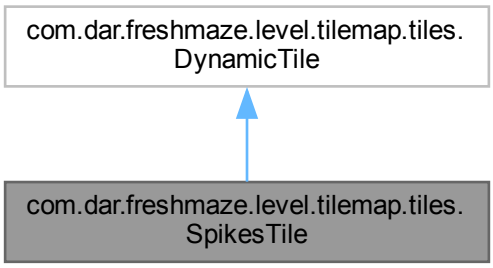
- core/src/com/dar/freshmaze/level/tilemap/SpikeGenerator.java

3.36 com.dar.freshmaze.level.tilemap.tiles.SpikesTile Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.tiles.SpikesTile:



Collaboration diagram for com.dar.freshmaze.level.tilemap.tiles.SpikesTile:



Public Member Functions

- **SpikesTile** (LevelTilemap tilemap, LevelTilemap.CellPos pos, TiledMapTile openTile)
- void **onTouch** (Bob bob)
- boolean **isOpen** ()
- void **setOpen** (boolean newIsOpen)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

- **DynamicTile** (LevelTilemap tilemap, LevelTilemap.CellPos cellPos, TiledMapTile defaultTile, Level↩ Tilemap.Layer defaultLayer)
- LevelTilemap **getTilemap** ()
- Body **getPhysBody** ()
- LevelTilemap.CellPos **getCellPos** ()
- TiledMapTile **getDefaultTile** ()
- LevelTilemap.Layer **getDefaultLayer** ()

Additional Inherited Members

Protected Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

- void **setPhysBody** (Body newPhysBody)

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/tilemap/tiles/SpikesTile.java

3.37 com.dar.freshmaze.entities.Entity.SpriteKind Enum Reference

Public Member Functions

- Matrix4 **getRenderMatrix** ()

Public Attributes

- **Isometric**
- **Transform**

The documentation for this enum was generated from the following file:

- core/src/com/dar/freshmaze/entities/Entity.java

3.38 com.dar.freshmaze.level.tilemap.tiles.EntranceTile.State Enum Reference

Public Attributes

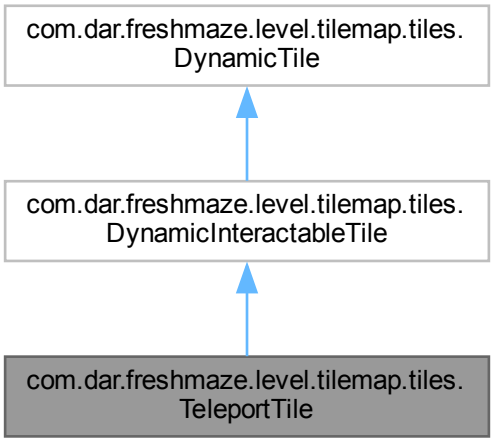
- **Open**
- **Closed**
- **Cleared**

The documentation for this enum was generated from the following file:

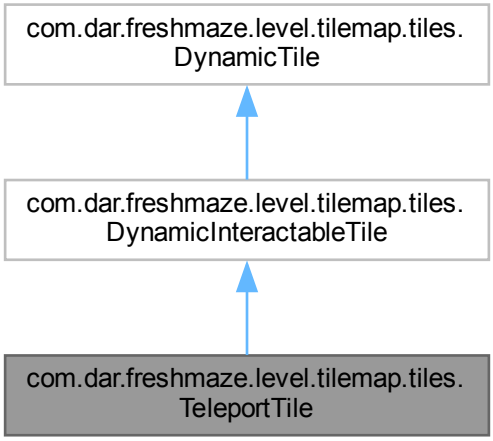
- core/src/com/dar/freshmaze/level/tilemap/tiles/EntranceTile.java

3.39 com.dar.freshmaze.level.tilemap.tiles.TeleportTile Class Reference

Inheritance diagram for com.dar.freshmaze.level.tilemap.tiles.TeleportTile:



Collaboration diagram for com.dar.freshmaze.level.tilemap.tiles.TeleportTile:



Public Member Functions

- **TeleportTile** (LevelTilemap tilemap, LevelTilemap.CellPos pos, TiledMapTile tile, Dungeon dungeon)
- void interact (Bob player)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicInteractableTile

- **DynamicInteractableTile** (LevelTilemap tilemap, LevelTilemap.CellPos pos, TiledMapTile defaultTile, LevelTilemap.Layer defaultLayer)
- abstract void **interact** (Bob player)

Public Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

- **DynamicTile** (LevelTilemap tilemap, LevelTilemap.CellPos cellPos, TiledMapTile defaultTile, LevelTilemap.Layer defaultLayer)
- LevelTilemap **getTilemap** ()
- Body **getPhysBody** ()
- LevelTilemap.CellPos **getCellPos** ()
- TiledMapTile **getDefaultTile** ()
- LevelTilemap.Layer **getDefaultLayer** ()

Additional Inherited Members

Protected Member Functions inherited from com.dar.freshmaze.level.tilemap.tiles.DynamicTile

- void **setPhysBody** (Body newPhysBody)

3.39.1 Member Function Documentation

3.39.1.1 interact()

```
void com.dar.freshmaze.level.tilemap.tiles.TeleportTile.interact (
            Bob player )  [inline]
```

Reimplemented from com.dar.freshmaze.level.tilemap.tiles.DynamicInteractableTile.

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/level/tilemap/tiles/TeleportTile.java

3.40 com.dar.freshmaze.util.TimeUtil Class Reference

Static Public Member Functions

- static void **init** ()
- static long **time** ()
- static float **timef** ()

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/util/TimeUtil.java

3.41 com.dar.freshmaze.ui.ScreenTransition.TransitionCallback Interface Reference

Public Member Functions

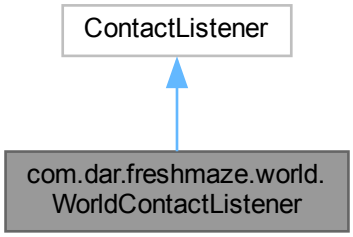
- void **onComplete** ()

The documentation for this interface was generated from the following file:

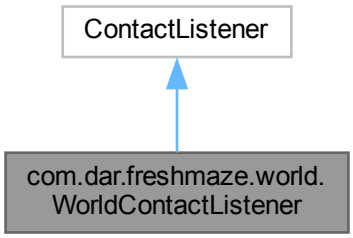
- core/src/com/dar/freshmaze/ui/ScreenTransition.java

3.42 com.dar.freshmaze.world.WorldContactListener Class Reference

Inheritance diagram for com.dar.freshmaze.world.WorldContactListener:



Collaboration diagram for com.dar.freshmaze.world.WorldContactListener:



Public Member Functions

- void **beginContact** (Contact contact)
- void **endContact** (Contact contact)
- void **preSolve** (Contact contact, Manifold oldManifold)
- void **postSolve** (Contact contact, ContactImpulse impulse)

The documentation for this class was generated from the following file:

- core/src/com/dar/freshmaze/world/WorldContactListener.java

6. Інструкція користувача

При запуску програми користувач бачить початкове вікно реалізованої програми:

Рис. 2: Головний екран

Гравець має натиснути пробіл, щоб розпочати гру. Далі він побачить свою початкову позицію. Наприклад:

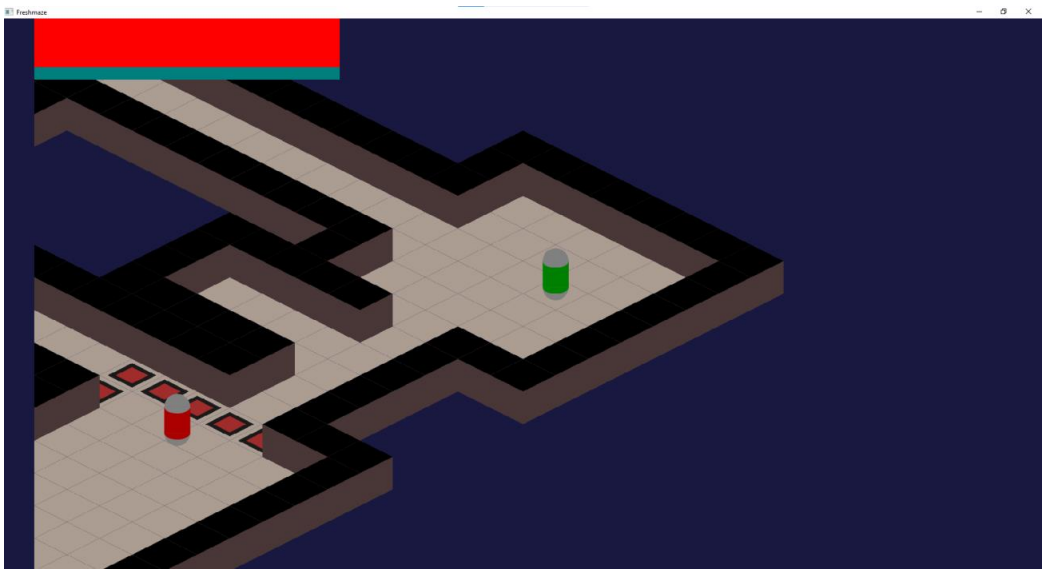


Рис. 3: Початок гри

Користувач може рухатися за допомогою таких клавіш:

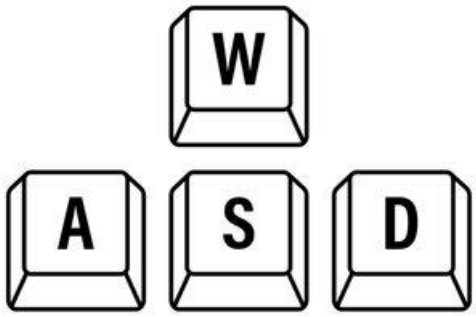


Рис. 4: Клавіші WASD

- W – рухатися вперед
- A – рухатися ліворуч
- S – рухатися назад
- D – рухатися праворуч

Також гравець може включити вільну камеру, щоб проглянути, на якій карті він знаходиться та знайти кімнату де портал. Щоб включити камеру треба натиснути на англійську клавішу P. Пересувати камеру можна за допомогою клавіш I,J,K,L. Ці клавіші є відповідними для таких: W,A,S,D:

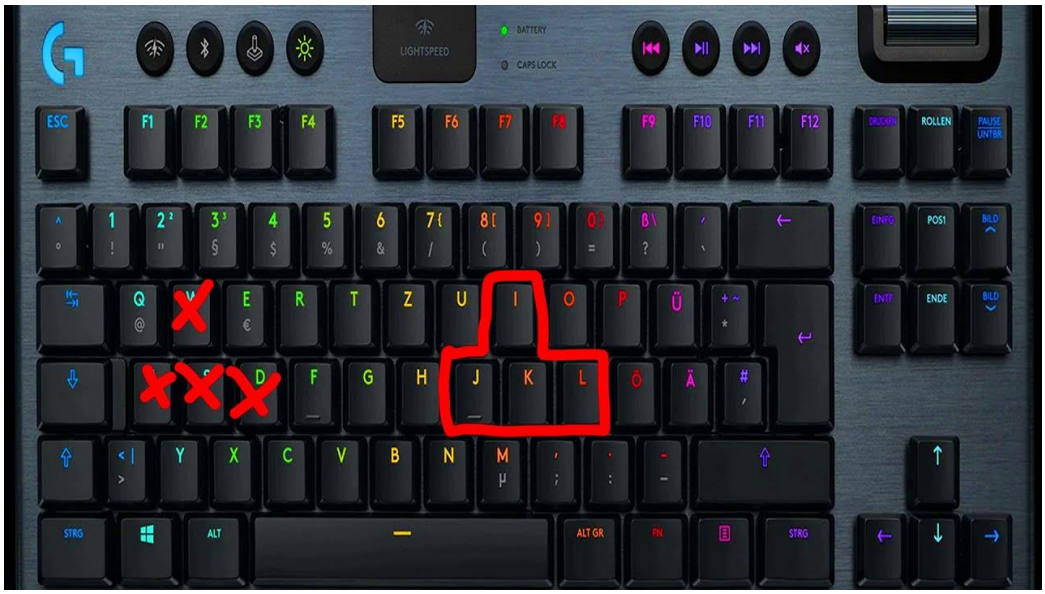


Рис. 5: Клавіші для управління грою

7. Проблеми, що виникали

1. Проблема з підтягуванням бібліотек та реалізації. Вирішення: вирішив все робити на Свінгу.
2. Проблема з великим вмістом інформації в екселі, при побудові та аналізі графіка функції. Вирішення: зменшив крок параметра функції.
3. Проблема відображення графіка. Вирішення: прочитання декількох онлайн ресурсів, що допомогло поправити код.

8. Висновки

Висновок: покращення навичок написання коду на Свінгу. А також здобуті навички пошуку інформації для способів вирішення різноманітних проблем.

Програмний код

FreshmazeGame.java

```
1 package com.dar.freshmaze;
2
3 import com.badlogic.gdx.Game;
4 import com.badlogic.gdx.Gdx;
5 import com.badlogic.gdx.audio.Sound;
6 import com.badlogic.gdx.graphics.OrthographicCamera;
```

```
7  import com.badlogic.gdx.graphics.g2d.SpriteBatch;
8  import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
9  import com.badlogic.gdx.utils.viewport.FitViewport;
10 import com.badlogic.gdx.utils.viewport.Viewport;
11 import com.dar.freshmaze.screens.GameScreen;
12 import com.dar.freshmaze.util.TimeUtil;
13
14 public class FreshmazeGame extends Game {
15   public final static float WIDTH = 1280;
16   public final static float HEIGHT = 720;
17
18   public SpriteBatch batch;
19   public ShapeRenderer shape;
20
21   @Override
22   public void create () {
23     TimeUtil.init();
24
25     batch = new SpriteBatch();
26     shape = new ShapeRenderer();
27     Sound sound = Gdx.audio.newSound(Gdx.files.internal("sound.mp3"));
28         sound.setLooping(sound.play(1.0f), true);
29
30     start(false);
31   }
32
33   public void start(boolean isRestart) {
34     if (getScreen() != null)
35       getScreen().dispose();
36
37     final OrthographicCamera camera = new OrthographicCamera();
38     final Viewport viewport = new FitViewport(WIDTH, HEIGHT, camera);
39
40     setScreen(new GameScreen(this, camera, viewport, isRestart));
41   }
42
43   @Override
44   public void dispose () {
45     batch.dispose();
46     shape.dispose();
47   }
48 }
```

GameScreen.java

```
1  package com.dar.freshmaze.screens;
2
3  import com.badlogic.gdx.Gdx;
```

```
4  import com.badlogic.gdx.Input;
5  import com.badlogic.gdx.Screen;
6  import com.badlogic.gdx.graphics.*;
7  import com.badlogic.gdx.graphics.g2d.BitmapFont;
8  import com.badlogic.gdx.graphics.g2d.TextureRegion;
9  import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
10 import com.badlogic.gdx.math.*;
11 import com.badlogic.gdx.physics.box2d.*;
12 import com.badlogic.gdx.scenes.scene2d.Stage;
13 import com.badlogic.gdx.scenes.scene2d.ui.Image;
14 import com.badlogic.gdx.scenes.scene2d.ui.Skin;
15 import com.badlogic.gdx.scenes.scene2d.ui.Table;
16 import com.badlogic.gdx.scenes.scene2d.ui.TextButton;
17 import com.badlogic.gdx.utils.Scaling;
18 import com.badlogic.gdx.utils.ScreenUtils;
19 import com.badlogic.gdx.utils.viewport.FitViewport;
20 import com.badlogic.gdx.utils.viewport.Viewport;
21 import com.dar.freshmaze.FreshmazeGame;
22 import com.dar.freshmaze.entities.Bob;
23 import com.dar.freshmaze.graphics.DepthSortedStage;
24 import com.dar.freshmaze.indicator.RectIndicator;
25 import com.dar.freshmaze.level.Dungeon;
26 import com.dar.freshmaze.level.Level;
27 import com.dar.freshmaze.level.tilemap.LevelTilemap;
28 import com.dar.freshmaze.ui.ScreenTransition;
29 import com.dar.freshmaze.util.IsometricUtil;
30 import com.dar.freshmaze.world.WorldContactListener;
31
32 public class GameScreen implements Screen {
33     private static final float CAMERA_ZOOM = 1.5f / 128.0f;
34
35     private static final float CAMERA_SPEED = 2.0f;
36     private static final float CAMERA_ZOOM_SPEED = 0.1f;
37
38     private final FreshmazeGame game;
39
40     private final OrthographicCamera camera;
41     private final Viewport viewport;
42
43     private final World physWorld;
44     private final Box2DDebugRenderer physDebugRenderer;
45     private boolean enableFreeCamera = false;
46
47     private final static int[] TILEMAP_FLOOR_LAYER = new int[] {
           LevelTilemap.Layer.Floor.getIndex(), LevelTilemap.Layer.
           FloorOverlay.getIndex() };
48     private final static int[] TILEMAP_WALL_LAYER = new int[] {
           LevelTilemap.Layer.Wall.getIndex() };
```

```
49
50      private final Skin skin = createSkin();
51
52      private final DepthSortedStage stage;
53      private final Dungeon dungeon;
54      private final Stage uiStage;
55      private final Bob bob;
56
57      private boolean mainInput = true;
58      private boolean hasGameBegun = false;
59      private boolean shouldRestart = false;
60
61      private LevelTransitionState levelTransitionState =
            LevelTransitionState.None;
62      private ScreenTransition levelChangeTransitionScreen;
63      private ScreenTransition startGameTransitionScreen;
64
65      public GameScreen(FreshmazeGame game, OrthographicCamera camera,
            Viewport viewport, boolean skipMenu) {
66          this.game = game;
67
68          this.camera = camera;
69          camera.zoom = CAMERA_ZOOM;
70          this.viewport = viewport;
71          hasGameBegun = skipMenu;
72
73          physWorld = new World(Vector2.Zero, true);
74          physWorld.setContactListener(new WorldContactListener());
75          physDebugRenderer = new Box2DDebugRenderer();
76
77          stage = new DepthSortedStage(viewport);
78          final Level level = new Level(physWorld, stage);
79
80          bob = new Bob(physWorld, level, Vector2.Zero);
81          stage.addActor(bob);
82
83          dungeon = new Dungeon(level, bob);
84
85          uiStage = new Stage(new FitViewport(game.WIDTH, game.HEIGHT));
86          if (!hasGameBegun) {
87              addGameStartUI();
88          } else {
89              uiStage.addActor(new ScreenTransition(1.0f, 3.0f, true, this
                    ::startGame));
90          }
91      }
92
93      private void createUI() {
```

```
94          final RectIndicator healthIndicator = new RectIndicator(new
                RectIndicator.FloatRangeBinder() {
95              @Override
96              public float getValue() {
97                  return dungeon.getBob().getHealth();
98              }
99
100             @Override
101             public float getMaxValue() {
102                 return dungeon.getBob().getMaxHealth();
103             }
104         });
105         healthIndicator.setIndicatorColor(Color.RED);
106         healthIndicator.setBounds(0.0f, uiStage.getHeight() - 64.0f,
                400.0f, 64.0f);
107
108         final RectIndicator attackIndicator = new RectIndicator(new
                RectIndicator.FloatRangeBinder() {
109             @Override
110             public float getValue() {
111                 return getMaxValue() - Math.max(dungeon.getBob().
                        getAttackTimeLeft(), 0.0f);
112             }
113
114             @Override
115             public float getMaxValue() {
116                 return dungeon.getBob().getTimePerAttack();
117             }
118         });
119         attackIndicator.setIndicatorColor(Color.TEAL);
120         attackIndicator.setBounds(0.0f, uiStage.getHeight() - 80.0f,
                400.0f, 16.0f);
121
122         uiStage.addActor(healthIndicator);
123         uiStage.addActor(attackIndicator);
124     }
125
126     @Override
127     public void show() {
128
129     }
130
131     @Override
132     public void render(float delta) {
133         if (shouldRestart) {
134             game.start(true);
135
136             return;
```

```
137         }
138
139         if(bob.getHealth() <= 0) {
140             stage.setKeyboardFocus(null);
141             gameover();
142         }
143         handleInput(delta);
144
145         if (dungeon.isMaxLevel()) {
146             victory();
147         }
148
149         if (dungeon.isPendingTransition()) {
150             if (levelTransitionState == LevelTransitionState.None) {
151                 levelTransitionState = LevelTransitionState.InProcess;
152
153                 levelChangeTransitionScreen = new ScreenTransition(2.0f,
                        2.5f, false, () -> {
154                     levelTransitionState = LevelTransitionState.
                            Completed;
155
156                     levelChangeTransitionScreen.remove();
157                     levelChangeTransitionScreen = new ScreenTransition
                            (2.0f, 2.5f, true);
158                     uiStage.addActor(levelChangeTransitionScreen);
159                 });
160                 uiStage.addActor(levelChangeTransitionScreen);
161             }
162         }
163
164         if (levelTransitionState != LevelTransitionState.InProcess) {
165             if (levelTransitionState == LevelTransitionState.Completed)
166                 levelTransitionState = LevelTransitionState.None;
167
168             dungeon.update(delta);
169         }
170
171         stage.act();
172         uiStage.act();
173
174         Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT | GL20.
                GL_DEPTH_BUFFER_BIT);
175         ScreenUtils.clear(0.1f, 0.1f, 0.25f, 1.0f);
176
177         if (!enableFreeCamera) {
178             final Vector2 playerPos = new Vector2(dungeon.getBob().getX
                    (), dungeon.getBob().getY());
179             camera.position.set(IsometricUtil.cartToIso(playerPos), 0.0f
```

```
                );
180             camera.zoom = CAMERA_ZOOM;
181         }
182         camera.update();
183 
184         stage.getViewport().apply();
185         final Level level = dungeon.getLevel();
186         level.render(camera, delta, TILEMAP_FLOOR_LAYER, false);
187         // level.debugRender(camera, delta, Level.DebugRender.ROOMS |
                Level.DebugRender.GRID);
188         // debugRenderBobCell();
189         level.render(camera, delta, TILEMAP_WALL_LAYER, true);
190 
191         // debugRenderAxes();
192 
193         final Rectangle viewBounds = dungeon.getLevel().
                getTilemapRenderer().getViewBounds();
194         stage.setVerticalViewBounds(new Vector2(viewBounds.y, viewBounds
                .height));
195         stage.shaderSetVerticalViewBounds();
196         stage.draw();
197 
198         Gdx.gl.glDisable(GL20.GL_DEPTH_TEST);
199         uiStage.getViewport().apply();
200         uiStage.draw();
201 
202         // Draw physics shapes in cartesian coordinates system (top-down
                view)
203         // physDebugRenderer.render(physWorld, camera.combined);
204 
205         // physDebugRenderer.render(physWorld, new Matrix4(camera.
                combined).mul(IsometricUtil.ISO_TRANSFORMATION_MATRIX));
206 
207         physWorld.step(1.0f / 60.0f, 6, 2);
208     }
209 
210     private void startGame() {
211         hasGameBegun = true;
212 
213         uiStage.clear();
214 
215         Gdx.input.setInputProcessor(stage);
216         stage.setKeyboardFocus(bob);
217 
218         createUI();
219     }
220 
221     private void debugRenderBobCell() {
```

```
222         final Vector2 bobPos = new Vector2(dungeon.getBob().getX() +
                dungeon.getBob().getWidth() / 2, dungeon.getBob().getY() +
                dungeon.getBob().getHeight() / 2);
223         final LevelTilemap tilemap = dungeon.getLevel().getTilemap();
224         final Vector2 cellPos = tilemap.vecToCellPosVec(bobPos);
225         final Vector2 worldPos = tilemap.cellPosToVec(cellPos);
226
227         game.shape.setTransformMatrix(IsometricUtil.
                ISO_TRANSFORMATION_MATRIX);
228         game.shape.setProjectionMatrix(camera.combined);
229         game.shape.begin(ShapeRenderer.ShapeType.Filled);
230         game.shape.setColor(Color.YELLOW);
231         game.shape.rect(worldPos.x, worldPos.y, tilemap.getTileSize(),
                tilemap.getTileSize());
232         game.shape.end();
233     }
234
235     private void debugRenderAxes() {
236         final Vector2 xAxis = IsometricUtil.cartToIso(Vector2.X);
237         final Vector2 yAxis = IsometricUtil.cartToIso(Vector2.Y);
238
239         game.shape.setTransformMatrix(new Matrix4().idt());
240         game.shape.setProjectionMatrix(camera.combined);
241         game.shape.begin(ShapeRenderer.ShapeType.Filled);
242
243         game.shape.setColor(Color.RED);
244         game.shape.rectLine(Vector2.Zero, xAxis.scl(128.0f), 0.1f);
245         game.shape.setColor(Color.GREEN);
246         game.shape.rectLine(Vector2.Zero, yAxis.scl(128.0f),  0.1f);
247
248         game.shape.setColor(Color.PURPLE);
249         game.shape.rectLine(Vector2.Zero, new Vector2(Vector2.X).scl
                (128.0f), 0.1f);
250         game.shape.setColor(Color.YELLOW);
251         game.shape.rectLine(Vector2.Zero, new Vector2(Vector2.Y).scl
                (128.0f), 0.1f);
252
253         game.shape.end();
254     }
255
256     private void handleInput(float dt) {
257         if (!hasGameBegun) {
258             if (Gdx.input.isKeyJustPressed(Input.Keys.SPACE))
259                 startGameTransitionScreen.setIsFrozen(false);
260
261             return;
262         }
263
```

```
264         if(mainInput) {
265             if (Gdx.input.isKeyJustPressed(Input.Keys.P))
266                 enableFreeCamera = !enableFreeCamera;
267 
268             if (enableFreeCamera) {
269                 if (Gdx.input.isKeyPressed(Input.Keys.MINUS))
270                     camera.zoom += 0.001;
271                 else if (Gdx.input.isKeyPressed(Input.Keys.EQUALS))
272                     camera.zoom -= 0.001;
273 
274                 camera.zoom = Math.max(0.01f, camera.zoom);
275 
276                 final float cameraSpeedMult = Gdx.input.isKeyPressed(
                        Input.Keys.SHIFT_LEFT) ? 2.0f : 1.0f;
277 
278                 final Vector2 cameraMovementVec = getInputMovementVec(
                        Input.Keys.J, Input.Keys.L, Input.Keys.K, Input.Keys
                        .I, CAMERA_SPEED);
279                 camera.translate(cameraMovementVec.x * cameraSpeedMult *
                         dt, cameraMovementVec.y * cameraSpeedMult * dt);
280             }
281         } else {
282             if (Gdx.input.isKeyJustPressed(Input.Keys.R))
283                 shouldRestart = true;
284             if (Gdx.input.isKeyJustPressed(Input.Keys.Q))
285                 Gdx.app.exit();
286         }
287     }
288 
289     private Vector2 getInputMovementVec(int leftKey, int rightKey, int
        downKey, int upKey, float speed) {
290         final float valueX = getInputAxisValue(leftKey, rightKey);
291         final float valueY = getInputAxisValue(downKey, upKey);
292 
293         return new Vector2(valueX, valueY)
294                 .nor()
295                 .scl(speed);
296     }
297 
298     private float getInputAxisValue(int keyNegative, int keyPositive) {
299         return Gdx.input.isKeyPressed(keyNegative) ? -1.0f :
300                 Gdx.input.isKeyPressed(keyPositive) ? 1.0f :
301                         0.0f;
302     }
303 
304     @Override
305     public void resize(int width, int height) {
306         viewport.update(width, height);
```

```
307
308         uiStage.getViewport().update(width, height);
309     }
310
311     @Override
312     public void pause() {
313
314     }
315
316     @Override
317     public void resume() {
318
319     }
320
321     @Override
322     public void hide() {
323
324     }
325
326     @Override
327     public void dispose() {
328         dungeon.dispose();
329         uiStage.dispose();
330     }
331
332     private void addGameStartUI() {
333         Gdx.input.setInputProcessor(stage);
334
335         startGameTransitionScreen = new ScreenTransition(1.0f, 3.0f,
                true, this::startGame);
336         startGameTransitionScreen.setIsFrozen(true);
337         uiStage.addActor(startGameTransitionScreen);
338
339         TextureRegion logoTexture = new TextureRegion(new Texture(Gdx.
                files.internal("freshmaze_logo.png")));
340         Table table = new Table();
341         table.setFillParent(true);
342         table.setPosition(0, 0);
343         uiStage.addActor(table);
344         Image logoImage = new Image(logoTexture);
345         logoImage.setScaling(Scaling.fillX);
346         table.add(logoImage).top().row();
347
348         TextureRegion infoTexture = new TextureRegion(new Texture(Gdx.
                files.internal("start_info.png")));
349         Image infoImage = new Image(infoTexture);
350         infoImage.setScaling(Scaling.fillX);
351         table.add(infoImage).row();
```

```
352     }
353
354     private void gameover() {
355         mainInput = false;
356         Gdx.input.setInputProcessor(stage);
357
358         uiStage.addActor(new ScreenTransition(1.0f, 3.0f, false, () -> {
359             Table table = new Table();
360             table.setFillParent(true);
361             table.setPosition(0, 0);
362             uiStage.addActor(table);
363             TextureRegion texture = new TextureRegion(new Texture(Gdx.
                    files.internal("gameover.png")));
364             Image image = new Image(texture);
365             image.setScaling(Scaling.fillX);
366             table.add(image).row();
367             table.columnDefaults(1);
368
369             final TextButton button = new TextButton("Press␣Q␣to␣quit",
                    skin);
370             table.add(button).fill().row();
371
372             final TextButton button2 = new TextButton("Press␣R␣to␣
                    restart", skin);
373             table.add(button2).fill().row();
374         }));
375     }
376     private void victory() {
377         mainInput = false;
378         Gdx.input.setInputProcessor(stage);
379
380         uiStage.addActor(new ScreenTransition(1.0f, 3.0f, false, () -> {
381             Table table = new Table();
382             table.setFillParent(true);
383             table.setPosition(0, 0);
384             uiStage.addActor(table);
385
386             TextureRegion texture = new TextureRegion(new Texture(Gdx.
                    files.internal("victory.png")));
387             Image image = new Image(texture);
388             image.setScaling(Scaling.fillX);
389             table.add(image).row();
390             table.columnDefaults(1);
391
392             final TextButton button = new TextButton("Press␣Q␣to␣quit",
                    skin);
393             table.add(button).fill().row();
394
```

```
395             final TextButton button2 = new TextButton("Press␣R␣to␣
                    restart", skin);
396             table.add(button2).fill().row();
397         }));
398     }
399
400     private static Skin createSkin() {
401         Skin skin = new Skin();
402
403         // Generate a 1x1 white texture and store it in the skin named "
                white".
404         Pixmap pixmap = new Pixmap(1, 1, Pixmap.Format.RGBA8888);
405         pixmap.setColor(Color.WHITE);
406         pixmap.fill();
407         skin.add("white", new Texture(pixmap));
408
409         // Store the default libGDX font under the name "default".
410         skin.add("default", new BitmapFont());
411
412         // Configure a TextButtonStyle and name it "default". Skin
                resources are stored by type, so this doesn't overwrite the
                font.
413         TextButton.TextButtonStyle textButtonStyle = new TextButton.
                TextButtonStyle();
414         textButtonStyle.up = skin.newDrawable("white", Color.DARK_GRAY);
415         textButtonStyle.down = skin.newDrawable("white", Color.DARK_GRAY
                );
416         textButtonStyle.checked = skin.newDrawable("white", Color.BLUE);
417         textButtonStyle.over = skin.newDrawable("white", Color.
                LIGHT_GRAY);
418         textButtonStyle.font = skin.getFont("default");
419         skin.add("default", textButtonStyle);
420
421         return skin;
422     }
423
424     private enum LevelTransitionState {
425         None,
426         InProcess,
427         Completed
428     }
429 }
```

Graph.java

```
1 package com.dar.freshmaze.common;
2
3 import java.util.Collections;
```

```
4  import java.util.HashMap;
5  import java.util.Map;
6  import java.util.Set;
7
8  public class Graph<VertexT, EdgeT> {
9      final HashMap<VertexT, HashMap<VertexT, EdgeT>> graph = new HashMap
           <>();
10
11     public Set<Map.Entry<VertexT, HashMap<VertexT, EdgeT>>> entrySet() {
12         return graph.entrySet();
13     }
14
15     public Map<VertexT, EdgeT> getConnections(VertexT vertex) {
16         final Map<VertexT, EdgeT> connections = graph.get(vertex);
17         if (connections == null)
18             return null;
19
20         return Collections.unmodifiableMap(connections);
21     }
22
23     public boolean contains(VertexT first) {
24         return graph.containsKey(first);
25     }
26
27     public void add(VertexT first, VertexT second, EdgeT edge) {
28         addDirected(first, second, edge);
29         addDirected(second, first, edge);
30     }
31
32     public void addDirected(VertexT first, VertexT second, EdgeT edge) {
33         final HashMap<VertexT, EdgeT> connections = graph.getOrDefault(
               first, new HashMap<>());
34         connections.put(second, edge);
35
36         graph.put(first, connections);
37     }
38
39     public void remove(VertexT vertex) {
40         if (!graph.containsKey(vertex))
41             return;
42
43         for (VertexT connectedVertex : graph.get(vertex).keySet())
44             graph.get(connectedVertex).remove(vertex);
45
46         graph.remove(vertex);
47     }
48 }
```

LevelBitmap.java

```java
package com.dar.freshmaze.level.bitmap;

import com.badlogic.gdx.math.Rectangle;
import com.badlogic.gdx.math.Vector2;
import com.dar.freshmaze.level.graph.LevelNodeGenerator;
import com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom;
import com.dar.freshmaze.level.tilemap.rooms.FinalLevelRoom;
import com.dar.freshmaze.util.RectangleUtil;

import java.util.ArrayList;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class LevelBitmap {
    private int width;
    private int height;

    private ArrayList<Cell> cells;

    private LevelNodeGenerator generator;

    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }

    public void generate(LevelNodeGenerator generator) {
        final Vector2 levelSize = generator.getLevelSize();

        this.generator = generator;

        width = (int)levelSize.x;
        height = (int)levelSize.y;

        cells = Stream
                .generate(() -> new Cell(Cell.Kind.Empty))
                .limit((long)width * height)
                .collect(Collectors.toCollection(ArrayList::new));

        placeHalls();
        placeRooms();

        // debugPrint();
    }
```

```
48
49      private void debugPrint() {
50          System.out.println("Width: " + width + " height: " + height);
51          for (int yi = height - 1; yi >= 0; --yi) {
52              StringBuilder line = new StringBuilder();
53              for (int xi = 0; xi < height; ++xi) {
54                  line.append(getDebugCellChar(getCell(xi, yi)));
55              }
56              System.out.println(line);
57          }
58      }
59
60      private char getDebugCellChar(Cell cell) {
61          switch (cell.kind) {
62              case HallEntrance:
63                  return 'E';
64              case Hall:
65                  return 'H';
66              case Wall:
67                  return 'W';
68              case Empty:
69                  return '_';
70              case Room:
71                  return 'R';
72              case Teleport:
73                  return 'T';
74              case Spikes:
75                  return 'S';
76          };
77
78          return '@';
79      }
80
81      private void placeRooms() {
82          generator.getRooms().forEach(room -> {
83              processRectangleMap(room.getBounds(), (kind, x, y) -> {
84                  switch (kind) {
85                      case Empty:
86                      case Hall:
87                      case Wall:
88                          return Cell.Kind.Room;
89                      default:
90                          return kind;
91                  }
92              });
93
94              if (room instanceof FinalLevelRoom) {
95                  final FinalLevelRoom finalRoom = (FinalLevelRoom)room;
```

```
96                  final Vector2 teleportPos = finalRoom.getTeleportPos();
97
98                  getCell((int)teleportPos.x, (int)teleportPos.y).setKind(
                        Cell.Kind.Teleport);
99              } else if (room instanceof BattleLevelRoom) {
100                 final BattleLevelRoom battleRoom = (BattleLevelRoom)room
                        ;
101
102                 battleRoom.getSpikes().forEach(pos ->
103                     getCell((int)pos.x, (int)pos.y).setKind(Cell.Kind.
                            Spikes)
104                 );
105             }
106         });
107
108         generator.getRooms().forEach(room ->
109                 processRectangleMap(RectangleUtil.expand(room.getBounds
                        (), new Vector2(1, 1)), (kind, x, y) -> {
110                     switch (kind) {
111                         case Empty:
112                             return Cell.Kind.Wall;
113                         case Hall:
114                             if (room instanceof BattleLevelRoom) {
115                                 final BattleLevelRoom battleRoom = (
                                        BattleLevelRoom)room;
116                                 battleRoom.addEntrance(new Vector2(x, y)
                                        );
117
118                                 return Cell.Kind.HallEntrance;
119                             } else {
120                                 return Cell.Kind.Hall;
121                             }
122                         default:
123                             return kind;
124                     }
125                 })
126         );
127     }
128
129     private void placeHalls() {
130         generator.getHalls().forEach(hall ->
131                 processRectangleMap(hall, (kind, x, y) -> {
132                     switch (kind) {
133                         case Empty:
134                             return Cell.Kind.Hall;
135                         default:
136                             return kind;
137                     }
```

```
138                 })
139         );
140 
141         generator.getHalls().forEach(hall ->
142                 processRectangleMap(RectangleUtil.expand(hall, new
                        Vector2(1, 1)), (kind, x, y) -> {
143                     switch (kind) {
144                         case Empty:
145                             return Cell.Kind.Wall;
146                         default:
147                             return kind;
148                     }
149                 })
150         );
151     }
152 
153     private void processRectangleMap(Rectangle rect, CellKindMapper
            cellKindMapper) {
154         processRectangle(rect, (cell, x, y) -> cell.setKind(
                cellKindMapper.processCell(cell.getKind(), x, y)));
155     }
156 
157     private void processRectangle(Rectangle rect, CellProcessor
            cellProcessor) {
158         for (int yi = 0; yi < rect.height; ++yi) {
159             for (int xi = 0; xi < rect.width; ++xi) {
160                 final int x = (int)rect.x + xi;
161                 final int y = (int)rect.y + yi;
162                 final Cell cell = getCell(x, y);
163 
164                 cellProcessor.processCell(cell, x, y);
165             }
166         }
167     }
168 
169     public Cell getCell(int xi, int yi) {
170         return getCell(xi + yi * width);
171     }
172 
173     public Cell getCell(int index) {
174         return cells.get(index);
175     }
176 
177     private interface CellKindMapper {
178         Cell.Kind processCell(Cell.Kind kind, int x, int y);
179     }
180 
181     private interface CellProcessor {
```

```
182         void processCell(Cell cell, int x, int y);
183     }
184
185
186     public static class Cell {
187         public enum Kind {
188             Empty,
189             Room,
190             Hall,
191             Wall,
192             HallEntrance,
193             Teleport,
194             Spikes
195         }
196
197         private Kind kind;
198
199         public Cell(Kind kind) {
200             this.kind = kind;
201         }
202
203         public Kind getKind() {
204             return kind;
205         }
206
207         public void setKind(Kind newKind) {
208             kind = newKind;
209         }
210     }
211 }
```

LevelGraph.java

```
1  package com.dar.freshmaze.level.graph;
2
3  import com.badlogic.gdx.math.Intersector;
4  import com.badlogic.gdx.math.Rectangle;
5  import com.badlogic.gdx.math.Vector2;
6  import com.dar.freshmaze.common.Graph;
7  import com.dar.freshmaze.util.RectangleUtil;
8
9  import java.util.HashMap;
10 import java.util.List;
11 import java.util.Map;
12 import java.util.Set;
13
14 /**
15  * Class that represents the graph of the levels.
```

```
16  */
17 public class LevelGraph {
18     private static final float EXPAND_DELTA = 1.0f;
19     private static final Vector2 EXPAND_HORIZONTAL = new Vector2(
           EXPAND_DELTA, 0.0f);
20     private static final Vector2 EXPAND_VERTICAL = new Vector2(0.0f,
           EXPAND_DELTA);
21
22     private Graph<LevelNode, Edge> graph;
23
24     public Set<Map.Entry<LevelNode, HashMap<LevelNode, Edge>>> entrySet
           () {
25         return graph.entrySet();
26     }
27
28     /**
29      * Generate level graph from leaves (list of level nodes)
30      * @param leaves the leaves
31      */
32     public void generate(List<LevelNode> leaves) {
33         graph = new Graph<>();
34
35         for (LevelNode firstNode : leaves) {
36             for (LevelNode secondNode : leaves) {
37                 if (firstNode.equals(secondNode))
38                     continue;
39
40                 final Edge edge = findIntersection(firstNode, secondNode
                       );
41                 if (edge == null)
42                     continue;
43
44                 graph.addDirected(firstNode, secondNode, edge);
45             }
46         }
47
48         // TODO: Create random breaks
49     }
50
51     private Edge findIntersection(LevelNode firstNode, LevelNode
           secondNode) {
52         final Rectangle intersection = new Rectangle();
53
54         final boolean isVertical;
55         if (intersectExpandedNodes(firstNode, secondNode,
               EXPAND_VERTICAL, intersection)) {
56             isVertical = true;
57         } else if (intersectExpandedNodes(firstNode, secondNode,
```

```
            EXPAND_HORIZONTAL, intersection)) {
58              isVertical = false;
59          } else {
60              return null;
61          }
62
63          return new Edge(intersection, isVertical);
64      }
65
66      private static boolean intersectExpandedNodes(LevelNode firstNode,
            LevelNode secondNode, Vector2 delta, Rectangle intersection) {
67          return Intersector.intersectRectangles(
68                  RectangleUtil.expand(firstNode.getBounds(), delta),
69                  RectangleUtil.expand(secondNode.getBounds(), delta),
70                  intersection
71          );
72      }
73
74      public static class Edge {
75          private final Rectangle intersection;
76          private final boolean isVertical;
77
78          public Edge(Rectangle intersection, boolean isVertical) {
79              this.intersection = intersection;
80              this.isVertical = isVertical;
81          }
82
83          public Rectangle getIntersection() {
84              return intersection;
85          }
86
87          public boolean isVertical() {
88              return isVertical;
89          }
90      }
91  }
```

LevelNode.java

```
1   package com.dar.freshmaze.level.graph;
2
3   import com.badlogic.gdx.math.MathUtils;
4   import com.badlogic.gdx.math.Rectangle;
5
6   public class LevelNode {
7       private final LevelNodeGenerationRules rules;
8       private final Rectangle bounds;
9       private LevelNode leftChild;
```

```
10      private LevelNode rightChild;
11      private Rectangle roomBounds;
12
13      public LevelNode(Rectangle bounds, LevelNodeGenerationRules rules) {
14          this.bounds = bounds;
15          this.rules = rules;
16      }
17
18      public boolean isLeaf() {
19          return leftChild == null && rightChild == null;
20      }
21
22      public Rectangle getBounds() {
23          return bounds;
24      }
25
26      public LevelNode getLeftChild() {
27          return leftChild;
28      }
29
30      public LevelNode getRightChild() {
31          return rightChild;
32      }
33
34      public Rectangle getRoomBounds() {
35          return roomBounds;
36      }
37
38      public boolean split() {
39          if (leftChild != null || rightChild != null)
40              return false;
41
42          if (bounds.width < rules.getMaxNodeSize() && bounds.height <
                rules.getMaxNodeSize() ) {
43              if (MathUtils.random() < rules.getSplitChance())
44                  return false;
45          }
46
47
48          final boolean isSplitVert = isSplitVertical();
49
50          final float maxSize = isSplitVert ? bounds.height : bounds.width
                ;
51          final float maxSplitSize = maxSize - rules.getMinNodeSize();
52          if (maxSplitSize <= rules.getMinNodeSize())
53              return false;
54
55          // NOTE: Casting to int is used to conform to the grid
```

```
56          final float splitSize = MathUtils.random(rules.getMinNodeSize(),
                (int)maxSplitSize);
57
58          if (isSplitVert) {
59              leftChild = new LevelNode(new Rectangle(bounds.x, bounds.y,
                    bounds.width, splitSize), rules);
60              rightChild = new LevelNode(new Rectangle(bounds.x, bounds.y
                    + splitSize, bounds.width, bounds.height - splitSize),
                    rules);
61          } else {
62              leftChild = new LevelNode(new Rectangle(bounds.x, bounds.y,
                    splitSize, bounds.height), rules);
63              rightChild = new LevelNode(new Rectangle(bounds.x +
                    splitSize, bounds.y, bounds.width - splitSize, bounds.
                    height), rules);
64          }
65
66          return true;
67      }
68
69      public void generateRoom() {
70          if (!isLeaf())
71              return;
72
73          final float width = MathUtils.random(rules.getMinRoomSize(), (
                int)bounds.width - 2 * rules.getRoomGap());
74          final float height = MathUtils.random(rules.getMinRoomSize(), (
                int)bounds.height - 2 * rules.getRoomGap());
75
76          final float x = MathUtils.random(rules.getRoomGap(), (int)(
                bounds.width - width - rules.getRoomGap()));
77          final float y = MathUtils.random(rules.getRoomGap(), (int)(
                bounds.height - height - rules.getRoomGap()));
78
79          roomBounds = new Rectangle(bounds.x + x, bounds.y + y, width,
                height);
80      }
81
82      private boolean isSplitVertical() {
83          final float aspectRatio = bounds.getAspectRatio();
84
85          return 1.0f / aspectRatio >= 1.25f ? true :
86                 aspectRatio >= 1.25f ? false :
87                 MathUtils.randomBoolean();
88      }
89  }
```

LevelNodeGenerationRules.java

```java
package com.dar.freshmaze.level.graph;

public class LevelNodeGenerationRules {
    private final int minRoomSize;

    private final int minNodeSize;
    private final int maxNodeSize;
    private final float splitChance;
    private final int roomGap;

    public LevelNodeGenerationRules(int minRoomSize, int minNodeSize,
        int maxNodeSize, float splitChance, int roomGap) {
        this.minRoomSize = minRoomSize;
        this.minNodeSize = minNodeSize;
        this.maxNodeSize = maxNodeSize;
        this.splitChance = splitChance;
        this.roomGap = roomGap;
    }

    public int getMinRoomSize() {
        return minRoomSize;
    }

    public int getMinNodeSize() {
        return minNodeSize;
    }

    public int getMaxNodeSize() {
        return maxNodeSize;
    }

    public float getSplitChance() {
        return splitChance;
    }

    public int getRoomGap() {
        return roomGap;
    }
}
```

LevelNodeGenerator.java

```java
package com.dar.freshmaze.level.graph;

import com.badlogic.gdx.math.MathUtils;
import com.badlogic.gdx.math.Rectangle;
import com.badlogic.gdx.math.Vector2;
```

```
6  import com.badlogic.gdx.utils.Array;
7  import com.dar.freshmaze.level.EnemyGenerator;
8  import com.dar.freshmaze.level.tilemap.SpikeGenerator;
9  import com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom;
10 import com.dar.freshmaze.level.tilemap.rooms.FinalLevelRoom;
11 import com.dar.freshmaze.level.tilemap.rooms.LevelRoom;
12 import com.dar.freshmaze.level.tilemap.rooms.SpawnLevelRoom;
13 import com.dar.freshmaze.util.RectangleUtil;
14
15 import java.util.*;
16 import java.util.stream.Collectors;
17 import java.util.stream.IntStream;
18
19 public class LevelNodeGenerator {
20     private Vector2 levelSize;
21     private int hallThickness;
22     private LevelNodeGenerationRules rules;
23
24     private LevelNode root;
25     private ArrayList<LevelNode> leaves;
26     private ArrayList<LevelRoom> rooms;
27     private LevelGraph graph;
28     private ArrayList<Rectangle> halls;
29
30     private LevelRoom spawnRoom;
31     private LevelRoom finalRoom;
32
33
34     public void generate(Vector2 levelSize, int hallThickness,
           LevelNodeGenerationRules rules, EnemyGenerator enemyGenerator,
           SpikeGenerator spikeGenerator) {
35         if (rooms != null)
36             rooms.forEach(LevelRoom::onDestroy);
37
38         this.levelSize = levelSize;
39         this.hallThickness = hallThickness;
40         this.rules = rules;
41
42         generateNodes();
43         generateGraph();
44         generateHalls();
45         generateRooms(enemyGenerator, spikeGenerator);
46     }
47
48     public Vector2 getLevelSize() {
49         return levelSize;
50     }
51
```

```
52      public LevelNode getRoot() {
53          return root;
54      }
55
56      public List<LevelNode> getLeaves() {
57          return Collections.unmodifiableList(leaves);
58      }
59
60      public List<LevelRoom> getRooms() {
61          return Collections.unmodifiableList(rooms);
62      }
63
64      public LevelGraph getGraph() {
65          return graph;
66      }
67
68      public List<Rectangle> getHalls() {
69          return halls;
70      }
71
72      public LevelRoom getSpawnRoom() {
73          return spawnRoom;
74      }
75
76      public LevelRoom getFinalRoom() {
77          return finalRoom;
78      }
79
80      private void generateRooms(EnemyGenerator enemyGenerator,
            SpikeGenerator spikeGenerator) {
81          final ArrayList<Integer> indices = IntStream.range(0, leaves.
                size())
82                  .boxed()
83                  .collect(Collectors.toCollection(ArrayList::new));
84          Collections.shuffle(indices);
85
86          if (indices.size() < 2)
87              throw new RuntimeException("Can't␣generate␣dungeon␣with␣less
                    ␣than␣two␣rooms");
88
89          spawnRoom = createSpawnRoom(leaves.get(indices.get(0)));
90          finalRoom = createFinalRoom(leaves.get(indices.get(1)));
91
92          rooms.add(spawnRoom);
93          rooms.add(finalRoom);
94
95          for (int i = 2; i < indices.size(); ++i)
96              rooms.add(createBattleRoom(leaves.get(indices.get(i)),
```

```
                enemyGenerator, spikeGenerator));
97      }
98
99      private SpawnLevelRoom createSpawnRoom(LevelNode node) {
100         return new SpawnLevelRoom(node.getRoomBounds());
101     }
102
103     private FinalLevelRoom createFinalRoom(LevelNode node) {
104         final Rectangle bounds = node.getRoomBounds();
105         final Vector2 teleportPos = new Vector2(bounds.x + (int)(0.5f *
                bounds.width), bounds.y + (int)(0.5f * bounds.height));;
106
107         return new FinalLevelRoom(bounds, teleportPos);
108     }
109
110     private BattleLevelRoom createBattleRoom(LevelNode node,
            EnemyGenerator enemyGenerator, SpikeGenerator spikeGenerator) {
111         final float spikeInterval = MathUtils.random(1.0f, 4.0f);
112
113         final BattleLevelRoom room = new BattleLevelRoom(node.
                getRoomBounds(), enemyGenerator, spikeInterval);
114
115         room.setSpikes(spikeGenerator.generateSpikes(room));
116
117         return room;
118     }
119
120     private void generateHalls() {
121         halls = new ArrayList<>();
122
123         final HashMap<LevelGraph.Edge, Vector2> intersectionPoints = new
                HashMap<>();
124
125         for (Map.Entry<LevelNode, HashMap<LevelNode, LevelGraph.Edge>>
                entry : graph.entrySet()) {
126             final LevelNode firstNode = entry.getKey();
127
128             for (Map.Entry<LevelNode, LevelGraph.Edge> connection :
                    entry.getValue().entrySet()) {
129                 final LevelNode secondNode = connection.getKey();
130                 final LevelGraph.Edge edge = connection.getValue();
131
132                 if (!intersectionPoints.containsKey(edge))
133                     intersectionPoints.put(edge,
                            getRandomEdgeIntersectionPoint(edge));
134
135                 final Vector2 intersectionPoint = intersectionPoints.get
                        (edge);
```

```
136
137                 halls.addAll(joinRoomsWithHalls(firstNode, secondNode,
                        intersectionPoint));
138             }
139         }
140     }
141
142     private ArrayList<Rectangle> joinRoomsWithHalls(LevelNode firstNode,
            LevelNode secondNode, Vector2 intersectionPoint) {
143         final Rectangle room = firstNode.getRoomBounds();
144
145         return joinPointsWithHalls(
146                 new Vector2(room.x + (int)(0.5f * room.width), room.y +
                        (int)(0.5f * room.height)),
147                 intersectionPoint,
148                 hallThickness
149         );
150     }
151
152     static ArrayList<Rectangle> joinPointsWithHalls(Vector2 start,
            Vector2 end, int thickness) {
153         final Vector2 diff = end.sub(start);
154
155         final ArrayList<Rectangle> rects = new ArrayList<>();
156
157         if (MathUtils.randomBoolean()) {
158             rects.add(RectangleUtil.normalize(new Rectangle(start.x,
                        start.y - 0.5f * thickness, diff.x, thickness)));
159             rects.add(RectangleUtil.normalize(new Rectangle(start.x +
                        diff.x - 0.5f * thickness, start.y, thickness, diff.y)))
                        ;
160         } else {
161             rects.add(RectangleUtil.normalize(new Rectangle(start.x -
                        0.5f * thickness, start.y, thickness, diff.y)));
162             rects.add(RectangleUtil.normalize(new Rectangle(start.x,
                        start.y + diff.y - 0.5f * thickness, diff.x, thickness))
                        );
163         }
164
165         return rects;
166     }
167
168     Vector2 getRandomEdgeIntersectionPoint(LevelGraph.Edge edge) {
169         final Rectangle rect = edge.getIntersection();
170
171         // TODO: Create some utility function like getIntCenter
172         return new Vector2(rect.x + (int)(0.5f * rect.width), rect.y + (
                int)(0.5f * rect.height));
```

```
173     }
174
175     private void generateGraph() {
176         graph = new LevelGraph();
177         graph.generate(leaves);
178     }
179
180     private void generateNodes() {
181         leaves = new ArrayList<>();
182         rooms = new ArrayList<>();
183         root = new LevelNode(new Rectangle(0.0f, 0.0f, levelSize.x,
                levelSize.y), rules);
184         generateNodeRecursive(root);
185     }
186
187     private void generateNodeRecursive(LevelNode node) {
188         if (!node.split()) {
189             node.generateRoom();
190             leaves.add(node);
191
192             return;
193         }
194
195         generateNodeRecursive(node.getLeftChild());
196         generateNodeRecursive(node.getRightChild());
197     }
198 }
```

LevelTilemap.java

```
1  package com.dar.freshmaze.level.tilemap;
2
3  import com.badlogic.gdx.Gdx;
4  import com.badlogic.gdx.graphics.Texture;
5  import com.badlogic.gdx.graphics.g2d.TextureRegion;
6  import com.badlogic.gdx.maps.MapLayers;
7  import com.badlogic.gdx.maps.objects.RectangleMapObject;
8  import com.badlogic.gdx.maps.tiled.TiledMap;
9  import com.badlogic.gdx.maps.tiled.TiledMapTile;
10 import com.badlogic.gdx.maps.tiled.TiledMapTileLayer;
11 import com.badlogic.gdx.maps.tiled.tiles.StaticTiledMapTile;
12 import com.badlogic.gdx.math.MathUtils;
13 import com.badlogic.gdx.math.Rectangle;
14 import com.badlogic.gdx.math.Vector2;
15 import com.badlogic.gdx.physics.box2d.*;
16 import com.badlogic.gdx.utils.Array;
17 import com.badlogic.gdx.utils.Disposable;
18 import com.badlogic.gdx.utils.ObjectMap;
```

```
19 import com.dar.freshmaze.level.Dungeon;
20 import com.dar.freshmaze.level.bitmap.LevelBitmap;
21 import com.dar.freshmaze.level.tilemap.tiles.EntranceTile;
22 import com.dar.freshmaze.level.tilemap.tiles.SpikesTile;
23 import com.dar.freshmaze.level.tilemap.tiles.TeleportTile;
24 import com.dar.freshmaze.level.tilemap.tiles.DynamicTile;
25
26 import java.util.Objects;
27
28 /**
29  * TileMap for levels.
30  */
31 public class LevelTilemap implements Disposable {
32     private final Texture tiles;
33
34     public final StaticTiledMapTile floorTile;
35     public final StaticTiledMapTile wallTile;
36     public final StaticTiledMapTile entranceOpenTile;
37     public final StaticTiledMapTile entranceClearedTile;
38     public final StaticTiledMapTile entranceClosedTile;
39     public final StaticTiledMapTile teleportMonolithTile;
40     public final StaticTiledMapTile chestClosedTile;
41     public final StaticTiledMapTile chestOpenTile;
42     public final StaticTiledMapTile spikesTile;
43
44     private final float tileSize;
45     private final int textureTileSize;
46     private TiledMap tilemap;
47     private final Array<Body> physBodies = new Array<>();
48     private final ObjectMap<CellPos, DynamicTile> dynamicTiles = new
           ObjectMap<>();
49
50     private final World physWorld;
51     private Dungeon dungeon;
52
53     /**
54      *
55      * @param physWorld
56      * @param tilesetPath
57      * @param tileSize
58      * @param textureTileSize
59      */
60     public LevelTilemap(World physWorld, String tilesetPath, float
           tileSize, int textureTileSize) {
61         this.physWorld = physWorld;
62         this.tileSize = tileSize;
63         this.textureTileSize = textureTileSize;
64         tiles = new Texture(Gdx.files.internal(tilesetPath));
```

```
65          final TextureRegion[][] splitTiles = TextureRegion.split(tiles,
                textureTileSize, textureTileSize);
66
67          floorTile = new StaticTiledMapTile(splitTiles[0][0]);
68          wallTile = new StaticTiledMapTile(splitTiles[0][1]);
69          entranceOpenTile = new StaticTiledMapTile(splitTiles[0][2]);
70          entranceClearedTile = new StaticTiledMapTile(splitTiles[0][3]);
71          entranceClosedTile = new StaticTiledMapTile(splitTiles[0][4]);
72          teleportMonolithTile = new StaticTiledMapTile(splitTiles[0][5]);
73          chestClosedTile = new StaticTiledMapTile(splitTiles[0][6]);
74          chestOpenTile = new StaticTiledMapTile(splitTiles[0][7]);
75          spikesTile = new StaticTiledMapTile(splitTiles[1][0]);
76
77          wallTile.getObjects().add(new RectangleMapObject());
78          entranceClosedTile.getObjects().add(new RectangleMapObject());
79          teleportMonolithTile.getObjects().add(new RectangleMapObject());
80          chestClosedTile.getObjects().add(new RectangleMapObject());
81          chestOpenTile.getObjects().add(new RectangleMapObject());
82          spikesTile.getObjects().add(new RectangleMapObject());
83      }
84
85      public float getTileSize() {
86          return tileSize;
87      }
88
89      public int getTextureTileSize() {
90          return textureTileSize;
91      }
92
93      public TiledMap getTilemap() {
94          return tilemap;
95      }
96
97      // TODO: Remove this duplicated code
98      public Vector2 cellPosToVec(Vector2 cellPos) {
99          return new Vector2(
100                 cellPos.x * tileSize,
101                 cellPos.y * tileSize
102         );
103     }
104
105     public Vector2 cellPosToVec(CellPos cellPos) {
106         return new Vector2(
107                 cellPos.x * tileSize,
108                 cellPos.y * tileSize
109         );
110     }
111
```

```
112     public CellPos vecToCellPos(Vector2 pos) {
113         return new CellPos(
114                 MathUtils.floor(pos.x / tileSize),
115                 MathUtils.floor(pos.y / tileSize)
116         );
117     }
118 
119     public Vector2 vecToCellPosVec(Vector2 pos) {
120         return new Vector2(
121                 MathUtils.floor(pos.x / tileSize),
122                 MathUtils.floor(pos.y / tileSize)
123         );
124     }
125     //
126 
127     public World getPhysWorld() {
128         return physWorld;
129     }
130 
131     public Array<Body> getPhysBodies() {
132         return physBodies;
133     }
134 
135     public void setDungeon(Dungeon newDungeon) {
136         dungeon = newDungeon;
137     }
138     public Dungeon getDungeon() { return dungeon; }
139 
140     public void generate(LevelBitmap bitmap) {
141         tilemap = new TiledMap();
142         physBodies.forEach(physWorld::destroyBody);
143         physBodies.clear();
144         dynamicTiles.forEach(entry -> {
145             final Body physBody = entry.value.getPhysBody();
146             if (physBody != null)
147                 physWorld.destroyBody(physBody);
148         });
149         dynamicTiles.clear();
150 
151         final MapLayers layers = tilemap.getLayers();
152         for (int i = 0; i < 3; ++i)
153             layers.add(createLayer(bitmap.getWidth(), bitmap.getHeight()
                    ));
154 
155         for (int yi = 0; yi < bitmap.getHeight(); ++yi) {
156             for (int xi = 0; xi < bitmap.getWidth(); ++xi) {
157                 final LevelBitmap.Cell bitmapCell = bitmap.getCell(xi,
                        yi);
```

```
158
159                 mapBitmapCellToTile(new CellPos(xi, yi), bitmapCell);
160             }
161         }
162     }
163
164     public DynamicTile getDynamicTile(CellPos cellPos) {
165         return dynamicTiles.get(cellPos);
166     }
167
168     private static PolygonShape rectangleToPhysPolygon(Rectangle rect) {
169         final PolygonShape polygon = new PolygonShape();
170         polygon.setAsBox(
171                 rect.width * 0.5f,
172                 rect.height * 0.5f,
173                 rect.getCenter(new Vector2()),
174                 0.0f
175         );
176
177         return polygon;
178     }
179
180     private TiledMapTileLayer createLayer(int width, int height) {
181         final TiledMapTileLayer layer = new TiledMapTileLayer(width,
                height, textureTileSize, textureTileSize / 2);
182         layer.setOffsetY(0.25f * textureTileSize);
183
184         return layer;
185     }
186
187     // TODO: Create more tiles
188     private void mapBitmapCellToTile(CellPos pos, LevelBitmap.Cell
            bitmapCell) {
189         switch (bitmapCell.getKind()) {
190             case Wall:
191                 placeStaticTile(pos, wallTile, Layer.Wall);
192                 break;
193
194             case Room:
195             case Hall:
196                 placeStaticTile(pos, floorTile, Layer.Floor);
197                 break;
198
199             case HallEntrance:
200                 placeStaticTile(pos, floorTile, Layer.Floor);
201                 placeDynamicTile(new EntranceTile(this, pos,
                        entranceOpenTile, entranceClosedTile,
                        entranceClearedTile));
```

```
202                 break;
203
204             case Teleport:
205                 placeStaticTile(pos, floorTile, Layer.Floor);
206                 placeDynamicTile(new TeleportTile(this, pos,
                        teleportMonolithTile, dungeon));
207                 break;
208
209             case Spikes:
210                 placeStaticTile(pos, floorTile, Layer.Floor);
211                 placeDynamicTile(new SpikesTile(this, pos, spikesTile));
212                 break;
213
214             default:
215                 break;
216         }
217     }
218
219     public void placeDynamicTile(DynamicTile dynamicTile) {
220         final CellPos pos = dynamicTile.getCellPos();
221         placeTile(pos, dynamicTile.getDefaultTile(), dynamicTile.
                getDefaultLayer());
222
223         dynamicTiles.put(pos, dynamicTile);
224     }
225
226     private void placeStaticTile(CellPos pos, TiledMapTile tile, Layer
            layerIndex) {
227         placeTile(pos, tile, layerIndex);
228
229         final Body physBody = createTilePhysBody(pos, tile);
230         if (physBody != null)
231             physBodies.add(physBody);
232     }
233
234     public void placeTile(CellPos pos, TiledMapTile tile, Layer
            layerIndex) {
235         final TiledMapTileLayer layer = tilemap.getLayers().getByType(
                TiledMapTileLayer.class).get(layerIndex.getIndex());
236         final TiledMapTileLayer.Cell cell = new TiledMapTileLayer.Cell()
                ;
237         cell.setTile(tile);
238         layer.setCell(pos.getX(), pos.getY(), cell);
239     }
240
241     public Body createTilePhysBody(CellPos pos, TiledMapTile tile) {
242         return createTilePhysBodyImpl(pos, tile, false);
243     }
```

```
244
245     public Body createTilePhysBodySensor(CellPos pos, TiledMapTile tile)
            {
246         return createTilePhysBodyImpl(pos, tile, true);
247     }
248
249     private Body createTilePhysBodyImpl(CellPos pos, TiledMapTile tile,
            boolean isSensor) {
250         if (tile == null)
251             return null;
252
253         final Vector2 worldPos = cellPosToVec(pos);
254         final Array<RectangleMapObject> objects = tile.getObjects().
                getByType(RectangleMapObject.class);
255         if (objects.isEmpty())
256             return null;
257
258         final Rectangle tileRect = objects.get(0).getRectangle();
259         final Rectangle rect = new Rectangle(
260                 tileRect.x + worldPos.x,
261                 tileRect.y + worldPos.y,
262                 tileRect.width * tileSize,
263                 tileRect.height * tileSize
264         );
265
266         final PolygonShape shape = rectangleToPhysPolygon(rect);
267
268         final BodyDef bd = new BodyDef();
269         bd.type = BodyDef.BodyType.StaticBody;
270         final Body body = physWorld.createBody(bd);
271
272         final FixtureDef fdef = new FixtureDef();
273         fdef.shape = shape;
274         fdef.isSensor = isSensor;
275         body.createFixture(fdef);
276
277         shape.dispose();
278
279         return body;
280     }
281
282     @Override
283     public void dispose() {
284         tiles.dispose();
285     }
286
287     public static class CellPos {
288         private final int x;
```

```
289         private final int y;
290 
291         public CellPos(int x, int y) {
292             this.x = x;
293             this.y = y;
294         }
295 
296         public int getX() {
297             return x;
298         }
299 
300         public int getY() {
301             return y;
302         }
303 
304         @Override
305         public boolean equals(Object obj) {
306             if (obj == this)
307                 return true;
308 
309             if (obj == null || getClass() != obj.getClass())
310                 return false;
311 
312             final CellPos other = (CellPos)obj;
313 
314             return x == other.x && y == other.y;
315         }
316 
317         @Override
318         public int hashCode() {
319             return Objects.hash(x, y);
320         }
321     }
322 
323     public enum Layer {
324         Floor(0),
325         FloorOverlay(1),
326         Wall(2);
327 
328         private final int index;
329 
330         public int getIndex() {
331             return index;
332         }
333 
334         Layer(int index) {
335             this.index = index;
336         }
```

```
337     }
338 }
```

ChestTile.java

```
1  package com.dar.freshmaze.level.tilemap.tiles;
2
3  import com.badlogic.gdx.maps.tiled.TiledMapTile;
4  import com.badlogic.gdx.math.MathUtils;
5  import com.dar.freshmaze.entities.Bob;
6  import com.dar.freshmaze.level.tilemap.LevelTilemap;
7
8  public class ChestTile extends DynamicInteractableTile {
9      private final TiledMapTile closedTile;
10     private final TiledMapTile openTile;
11
12     private boolean isOpen = false;
13
14     public ChestTile(LevelTilemap tilemap, LevelTilemap.CellPos pos,
           TiledMapTile closedTile, TiledMapTile openTile) {
15         super(tilemap, pos, closedTile, LevelTilemap.Layer.Wall);
16
17         this.closedTile = closedTile;
18         this.openTile = openTile;
19     }
20
21     @Override
22     public void interact(Bob player) {
23         if (isOpen)
24             return;
25
26         final float random = MathUtils.random();
27         if (random < 0.2) {
28             player.heal(10);
29         } else if (random < 0.8) {
30             player.increaseAttackSpeed(0.15f);
31         }
32
33         isOpen = true;
34
35         getTilemap().placeTile(getCellPos(), openTile, LevelTilemap.
               Layer.Wall);
36         setPhysBody(getTilemap().createTilePhysBody(getCellPos(),
               openTile));
37     }
38 }
```

DynamicTile.java

```java
package com.dar.freshmaze.level.tilemap.tiles;

import com.badlogic.gdx.maps.tiled.TiledMapTile;
import com.badlogic.gdx.physics.box2d.Body;
import com.dar.freshmaze.level.tilemap.LevelTilemap;

public abstract class DynamicTile {
    private final LevelTilemap tilemap;
    private final LevelTilemap.CellPos cellPos;
    private final TiledMapTile defaultTile;
    private final LevelTilemap.Layer defaultLayer;

    private Body physBody;

    public DynamicTile(LevelTilemap tilemap, LevelTilemap.CellPos
        cellPos, TiledMapTile defaultTile, LevelTilemap.Layer
        defaultLayer) {
        this.tilemap = tilemap;
        this.cellPos = cellPos;
        this.defaultTile = defaultTile;
        this.defaultLayer = defaultLayer;
    }

    public LevelTilemap getTilemap() {
        return tilemap;
    }

    public Body getPhysBody() {
        return physBody;
    }

    public LevelTilemap.CellPos getCellPos() {
         return cellPos;
    }

    public TiledMapTile getDefaultTile() {
        return defaultTile;
    }

    public LevelTilemap.Layer getDefaultLayer() {
        return defaultLayer;
    }

    protected void setPhysBody(Body newPhysBody) {
        if (physBody != null)
            tilemap.getPhysWorld().destroyBody(physBody);

        physBody = newPhysBody;
```

```
47          if (physBody != null)
48              physBody.setUserData(this);
49      }
50  }
```

DynamicInteractableTile.java

```
1   package com.dar.freshmaze.level.tilemap.tiles;
2
3   import com.badlogic.gdx.maps.tiled.TiledMapTile;
4   import com.dar.freshmaze.entities.Bob;
5   import com.dar.freshmaze.level.tilemap.LevelTilemap;
6
7   public abstract class DynamicInteractableTile extends DynamicTile {
8       public DynamicInteractableTile(LevelTilemap tilemap, LevelTilemap.
            CellPos pos, TiledMapTile defaultTile, LevelTilemap.Layer
            defaultLayer) {
9           super(tilemap, pos, defaultTile, defaultLayer);
10
11          setPhysBody(tilemap.createTilePhysBody(getCellPos(), defaultTile
                ));
12      }
13
14      public abstract void interact(Bob player);
15  }
```

EntranceTile.java

```
1   package com.dar.freshmaze.level.tilemap.tiles;
2
3   import com.badlogic.gdx.maps.tiled.TiledMapTile;
4   import com.dar.freshmaze.level.tilemap.LevelTilemap;
5
6   public class EntranceTile extends DynamicTile {
7       private final TiledMapTile openTile;
8       private final TiledMapTile closedTile;
9       private final TiledMapTile clearedTile;
10
11      private State state = State.Closed;
12
13      public EntranceTile(LevelTilemap tilemap, LevelTilemap.CellPos pos,
            TiledMapTile openTile, TiledMapTile closedTile, TiledMapTile
            clearedTile) {
14          super(tilemap, pos, openTile, LevelTilemap.Layer.FloorOverlay);
15
16          this.openTile = openTile;
17          this.closedTile = closedTile;
18          this.clearedTile = clearedTile;
19
```

```
20          setPhysBody(tilemap.createTilePhysBody(getCellPos(),
                getDefaultTile()));
21      }
22
23      public State getState() {
24          return state;
25      }
26
27      public void setState(State newState) {
28          state = newState;
29
30          final LevelTilemap tilemap = getTilemap();
31
32          switch (state) {
33              case Open:
34                  tilemap.placeTile(getCellPos(), openTile, LevelTilemap.
                        Layer.FloorOverlay);
35                  tilemap.placeTile(getCellPos(), null, LevelTilemap.Layer
                        .Wall);
36                  setPhysBody(null);
37                  break;
38
39              case Cleared:
40                  tilemap.placeTile(getCellPos(), clearedTile,
                        LevelTilemap.Layer.FloorOverlay);
41                  tilemap.placeTile(getCellPos(), null, LevelTilemap.Layer
                        .Wall);
42                  setPhysBody(null);
43                  break;
44
45              case Closed:
46                  tilemap.placeTile(getCellPos(), null, LevelTilemap.Layer
                        .FloorOverlay);
47                  tilemap.placeTile(getCellPos(), closedTile, LevelTilemap
                        .Layer.Wall);
48                  setPhysBody(tilemap.createTilePhysBody(getCellPos(),
                        closedTile));
49                  break;
50          }
51      }
52
53      public enum State {
54          Open,
55          Closed,
56          Cleared
57      }
58  }
```

SpikesTile.java

```java
package com.dar.freshmaze.level.tilemap.tiles;

import com.badlogic.gdx.maps.tiled.TiledMapTile;
import com.badlogic.gdx.physics.box2d.Body;
import com.dar.freshmaze.entities.Bob;
import com.dar.freshmaze.level.tilemap.LevelTilemap;

public class SpikesTile extends DynamicTile {
    private final TiledMapTile openTile;

    private boolean isOpen = false;

    public SpikesTile(LevelTilemap tilemap, LevelTilemap.CellPos pos,
        TiledMapTile openTile) {
        super(tilemap, pos, null, LevelTilemap.Layer.FloorOverlay);

        this.openTile = openTile;
    }

    public void onTouch(Bob bob) {
        bob.damage(3);
    }

    public boolean isOpen() {
        return isOpen;
    }

    public void setOpen(boolean newIsOpen) {
        if (isOpen == newIsOpen)
            return;

        isOpen = newIsOpen;

        final TiledMapTile tile = isOpen ? openTile : null;
        final Body body = getTilemap().createTilePhysBodySensor(
            getCellPos(), tile);

        getTilemap().placeTile(getCellPos(), tile, LevelTilemap.Layer.
            FloorOverlay);
        setPhysBody(body);
    }
}
```

TeleportTile.java

```java
package com.dar.freshmaze.level.tilemap.tiles;

```

```java
import com.badlogic.gdx.maps.tiled.TiledMapTile;
import com.dar.freshmaze.entities.Bob;
import com.dar.freshmaze.level.Dungeon;
import com.dar.freshmaze.level.tilemap.LevelTilemap;

public class TeleportTile extends DynamicInteractableTile {
    private final Dungeon dungeon;

    private boolean wasActivated = false;

    public TeleportTile(LevelTilemap tilemap, LevelTilemap.CellPos pos,
        TiledMapTile tile, Dungeon dungeon) {
        super(tilemap, pos, tile, LevelTilemap.Layer.Wall);

        this.dungeon = dungeon;
    }

    @Override
    public void interact(Bob player) {
        if (wasActivated)
            return;

        wasActivated = true;

        dungeon.moveToNextLevel();
    }
}
```

BattleLevelRoom.java

```java
package com.dar.freshmaze.level.tilemap.rooms;

import com.badlogic.gdx.math.Rectangle;
import com.badlogic.gdx.math.Vector2;
import com.badlogic.gdx.utils.Array;
import com.dar.freshmaze.entities.Bob;
import com.dar.freshmaze.entities.Enemy;
import com.dar.freshmaze.entities.Entity;
import com.dar.freshmaze.level.EnemyGenerator;
import com.dar.freshmaze.level.tilemap.LevelTilemap;
import com.dar.freshmaze.level.tilemap.tiles.ChestTile;
import com.dar.freshmaze.level.tilemap.tiles.EntranceTile;
import com.dar.freshmaze.level.tilemap.tiles.DynamicTile;
import com.dar.freshmaze.level.tilemap.tiles.SpikesTile;

/**
 * Battle room class
 */
```

```
19 public class BattleLevelRoom extends LevelRoom {
20     private final Array<Vector2> entrances = new Array<>();
21     private Array<Vector2> spikes;
22     private final Array<Enemy> enemies;
23     private final Array<Entity> otherEntities;
24
25     private boolean wasEntered = false;
26     private boolean isCleared = false;
27
28     private final static float startSpikeInterval = 3.0f;
29     private final static float spikeActiveInterval = 1.5f;
30     private float spikeInterval;
31     private float spikeTimeLeft;
32     private boolean spikesActive = false;
33
34     public BattleLevelRoom(Rectangle bounds, EnemyGenerator
           enemyGenerator, float spikeInterval) {
35         super(bounds);
36
37         this.spikeInterval = spikeInterval;
38         this.spikeTimeLeft = startSpikeInterval;
39
40         final EnemyGenerator.Result result = enemyGenerator.generate(
               this);
41         enemies = result.enemies;
42         otherEntities = result.otherEntities;
43     }
44
45     public Array<Vector2> getEntrances() {
46         return entrances;
47     }
48
49     /**
50      * Add entrance to the room
51      * @param entrance
52      */
53     public void addEntrance(Vector2 entrance) {
54         entrances.add(entrance);
55     }
56
57     public Array<Vector2> getSpikes() { return spikes; }
58
59     public void setSpikes(Array<Vector2> newSpikes) {
60         spikes = newSpikes;
61     }
62
63     @Override
64     public void act(float dt) {
```

```
65          if (wasEntered && !isCleared) {
66              if (spikeTimeLeft < 0) {
67                  if (spikesActive) {
68                      spikeTimeLeft = spikeInterval;
69                      spikesActive = false;
70
71                      setSpikesOpen(false);
72                  } else {
73                      spikeTimeLeft = spikeActiveInterval;
74                      spikesActive = true;
75
76                      setSpikesOpen(true);
77                  }
78              }
79
80              spikeTimeLeft -= dt;
81          }
82      }
83
84      @Override
85      public void onDestroy() {
86          enemies.forEach(Enemy::destroy);
87          otherEntities.forEach(Entity::destroy);
88      }
89
90      @Override
91      public void onPlayerEnter(Bob bob) {
92          if (!isCleared) {
93              wasEntered = true;
94
95              setEntrancesState(EntranceTile.State.Closed);
96          }
97      }
98
99      public void onEnemyDeath(Enemy enemy) {
100         enemies.removeValue(enemy, true);
101
102         if (enemies.isEmpty()) {
103             isCleared = true;
104
105             onCleared();
106         }
107     }
108
109     private void onCleared() {
110         setEntrancesState(EntranceTile.State.Cleared);
111
112         final LevelTilemap tilemap = getLevel().getTilemap();
```

```
113         final LevelTilemap.CellPos pos = tilemap.vecToCellPos(getBounds
                ().getCenter(new Vector2()));
114         tilemap.placeDynamicTile(new ChestTile(tilemap, pos, tilemap.
                chestClosedTile, tilemap.chestOpenTile));
115
116         setSpikesOpen(false);
117     }
118
119     private void setEntrancesState(EntranceTile.State state) {
120         entrances.forEach(pos -> dynamicTileApply(pos, EntranceTile.
                class, tile -> tile.setState(state)));
121     }
122
123     private void setSpikesOpen(boolean isOpen) {
124         spikes.forEach(pos -> dynamicTileApply(pos, SpikesTile.class,
                tile -> tile.setOpen(isOpen)));
125     }
126
127     private <T extends DynamicTile> void dynamicTileApply(Vector2 pos,
            Class<T> type, DynamicTileAction<T> action) {
128         final T tile = getDynamicTile(pos, type);
129         if (tile == null)
130             return;
131
132         action.apply(tile);
133     }
134
135     @SuppressWarnings("unchecked")
136     private <T extends DynamicTile> T getDynamicTile(Vector2 pos, Class<
            T> type) {
137         final DynamicTile dynamicTile = getLevel().getTilemap().
                getDynamicTile(new LevelTilemap.CellPos((int)pos.x, (int)pos
                .y));
138         if (dynamicTile == null)
139             return null;
140
141         if (type.isInstance(dynamicTile))
142             return (T)dynamicTile;
143
144         return null;
145     }
146
147     interface DynamicTileAction<T extends DynamicTile> {
148         void apply(T tile);
149     }
150 }
```

FinalLevelRoom.java

```java
package com.dar.freshmaze.level.tilemap.rooms;

import com.badlogic.gdx.math.Rectangle;
import com.badlogic.gdx.math.Vector2;

public class FinalLevelRoom extends LevelRoom {
    private final Vector2 teleportPos;

    public FinalLevelRoom(Rectangle bounds, Vector2 teleportPos) {
        super(bounds);

        this.teleportPos = teleportPos;
    }

    public final Vector2 getTeleportPos() {
        return teleportPos;
    }
}
```

LevelRoom.java

```java
package com.dar.freshmaze.level.tilemap.rooms;

import com.badlogic.gdx.math.Rectangle;
import com.dar.freshmaze.entities.Bob;
import com.dar.freshmaze.level.Level;

public class LevelRoom {
    private Level level;
    private final Rectangle bounds;


    public LevelRoom(Rectangle bounds) {
        this.bounds = bounds;
    }

    public Level getLevel() {
        return level;
    }

    public void setLevel(Level newLevel) {
        level = newLevel;
    }

    public Rectangle getBounds() {
        return bounds;
    }
```

```
27
28
29      public void act(float dt) {}
30      public void onDestroy() {}
31
32      public void onPlayerEnter(Bob bob) {}
33      public void onPlayerExit(Bob bob) {}
34  }
```

SpawnLevelRoom.java

```
1  package com.dar.freshmaze.level.tilemap.rooms;
2
3  import com.badlogic.gdx.math.Rectangle;
4
5  public class SpawnLevelRoom extends LevelRoom {
6      public SpawnLevelRoom(Rectangle bounds) {
7          super(bounds);
8      }
9  }
```

SortedIsometricTiledMapRenderer.java

```
1  /*
       ******************************************************************************
2   * Copyright 2013 See AUTHORS file.
3   *
4   * Licensed under the Apache License, Version 2.0 (the "License");
5   * you may not use this file except in compliance with the License.
6   * You may obtain a copy of the License at
7   *
8   *   http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to in writing, software
11  * distributed under the License is distributed on an "AS IS" BASIS,
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
        implied.
13  * See the License for the specific language governing permissions and
14  * limitations under the License.
15  ******************************************************************************
        */
16
17 package com.dar.freshmaze.level.tilemap;
18
19 import static com.badlogic.gdx.graphics.g2d.Batch.*;
20
21 import com.badlogic.gdx.Gdx;
22 import com.badlogic.gdx.graphics.Color;
```

```
23 import com.badlogic.gdx.graphics.GL20;
24 import com.badlogic.gdx.graphics.g2d.Batch;
25 import com.badlogic.gdx.graphics.g2d.TextureRegion;
26 import com.badlogic.gdx.graphics.glutils.ShaderProgram;
27 import com.badlogic.gdx.maps.tiled.TiledMap;
28 import com.badlogic.gdx.maps.tiled.TiledMapTile;
29 import com.badlogic.gdx.maps.tiled.TiledMapTileLayer;
30 import com.badlogic.gdx.maps.tiled.TiledMapTileLayer.Cell;
31 import com.badlogic.gdx.maps.tiled.renderers.BatchTiledMapRenderer;
32 import com.badlogic.gdx.math.Matrix4;
33 import com.badlogic.gdx.math.Vector2;
34 import com.badlogic.gdx.math.Vector3;
35
36 /**
37  * Isometric tilemap renderer that writes to the depth buffer if {@link
        #writeDepth} flag is enabled
38  */
39 public class SortedIsometricTiledMapRenderer extends
       BatchTiledMapRenderer {
40     private boolean writeDepth = false;
41
42     private Matrix4 isoTransform;
43     private Matrix4 invIsotransform;
44     private Vector3 screenPos = new Vector3();
45
46     private Vector2 topRight = new Vector2();
47     private Vector2 bottomLeft = new Vector2();
48     private Vector2 topLeft = new Vector2();
49     private Vector2 bottomRight = new Vector2();
50
51     public SortedIsometricTiledMapRenderer (TiledMap map) {
52         super(map);
53         init();
54     }
55
56     public SortedIsometricTiledMapRenderer (TiledMap map, Batch batch) {
57         super(map, batch);
58         init();
59     }
60
61     public SortedIsometricTiledMapRenderer (TiledMap map, float
           unitScale) {
62         super(map, unitScale);
63         init();
64     }
65
66     public SortedIsometricTiledMapRenderer (TiledMap map, float
           unitScale, Batch batch) {
```

```
67          super(map, unitScale, batch);
68          init();
69      }
70
71      private void init () {
72          // create the isometric transform
73          isoTransform = new Matrix4();
74          isoTransform.idt();
75
76          // isoTransform.translate(0, 32, 0);
77          isoTransform.scale((float)(Math.sqrt(2.0) / 2.0), (float)(Math.
                sqrt(2.0) / 4.0), 1.0f);
78          isoTransform.rotate(0.0f, 0.0f, 1.0f, -45);
79
80          // ... and the inverse matrix
81          invIsotransform = new Matrix4(isoTransform);
82          invIsotransform.inv();
83
84          batch.setShader(createShader());
85      }
86
87      private Vector3 translateScreenToIso (Vector2 vec) {
88          screenPos.set(vec.x, vec.y, 0);
89          screenPos.mul(invIsotransform);
90
91          return screenPos;
92      }
93
94      public void setWriteDepth(boolean newWriteDepth) {
95          writeDepth = newWriteDepth;
96      }
97
98      @Override
99      public void renderTileLayer (TiledMapTileLayer layer) {
100         if (writeDepth) {
101             Gdx.gl.glEnable(GL20.GL_DEPTH_TEST);
102             Gdx.gl.glDepthMask(true);
103             Gdx.gl.glDepthFunc(GL20.GL_ALWAYS);
104         }
105
106         final Color batchColor = batch.getColor();
107         final float color = Color.toFloatBits(batchColor.r, batchColor.g
                , batchColor.b, batchColor.a * layer.getOpacity());
108
109         float tileWidth = layer.getTileWidth() * unitScale;
110         float tileHeight = layer.getTileHeight() * unitScale;
111
112         final float layerOffsetX = layer.getRenderOffsetX() * unitScale;
```

```
113         // offset in tiled is y down, so we flip it
114         final float layerOffsetY = -layer.getRenderOffsetY() * unitScale
                ;
115
116         float halfTileWidth = tileWidth * 0.5f;
117         float halfTileHeight = tileHeight * 0.5f;
118
119         // setting up the screen points
120         // COL1
121         topRight.set(viewBounds.x + viewBounds.width - layerOffsetX,
                viewBounds.y - layerOffsetY);
122         // COL2
123         bottomLeft.set(viewBounds.x - layerOffsetX, viewBounds.y +
                viewBounds.height - layerOffsetY);
124         // ROW1
125         topLeft.set(viewBounds.x - layerOffsetX, viewBounds.y -
                layerOffsetY);
126         // ROW2
127         bottomRight.set(viewBounds.x + viewBounds.width - layerOffsetX,
                viewBounds.y + viewBounds.height - layerOffsetY);
128
129         // transforming screen coordinates to iso coordinates
130         int row1 = (int)(translateScreenToIso(topLeft).y / tileWidth) -
                2;
131         int row2 = (int)(translateScreenToIso(bottomRight).y / tileWidth
                ) + 2;
132
133         int col1 = (int)(translateScreenToIso(bottomLeft).x / tileWidth)
                 - 2;
134         int col2 = (int)(translateScreenToIso(topRight).x / tileWidth) +
                 2;
135
136         for (int row = row2; row >= row1; row--) {
137             for (int col = col1; col <= col2; col++) {
138                 float x = (col * halfTileWidth) + (row * halfTileWidth);
139                 float y = (row * halfTileHeight) - (col * halfTileHeight
                        );
140
141                 final TiledMapTileLayer.Cell cell = layer.getCell(col,
                        row);
142                 if (cell == null) continue;
143                 final TiledMapTile tile = cell.getTile();
144
145                 if (tile != null) {
146                     final boolean flipX = cell.getFlipHorizontally();
147                     final boolean flipY = cell.getFlipVertically();
148                     final int rotations = cell.getRotation();
149
```

```
150                     TextureRegion region = tile.getTextureRegion();
151
152                     float x1 = x + tile.getOffsetX() * unitScale +
                            layerOffsetX;
153                     float y1 = y + tile.getOffsetY() * unitScale +
                            layerOffsetY;
154                     float x2 = x1 + region.getRegionWidth() * unitScale;
155                     float y2 = y1 + region.getRegionHeight() * unitScale
                            ;
156
157                     float u1 = region.getU();
158                     float v1 = region.getV2();
159                     float u2 = region.getU2();
160                     float v2 = region.getV();
161
162                     vertices[X1] = x1;
163                     vertices[Y1] = y1;
164                     vertices[C1] = color;
165                     vertices[U1] = u1;
166                     vertices[V1] = v1;
167
168                     vertices[X2] = x1;
169                     vertices[Y2] = y2;
170                     vertices[C2] = color;
171                     vertices[U2] = u1;
172                     vertices[V2] = v2;
173
174                     vertices[X3] = x2;
175                     vertices[Y3] = y2;
176                     vertices[C3] = color;
177                     vertices[U3] = u2;
178                     vertices[V3] = v2;
179
180                     vertices[X4] = x2;
181                     vertices[Y4] = y1;
182                     vertices[C4] = color;
183                     vertices[U4] = u2;
184                     vertices[V4] = v1;
185
186                     if (flipX) {
187                         float temp = vertices[U1];
188                         vertices[U1] = vertices[U3];
189                         vertices[U3] = temp;
190                         temp = vertices[U2];
191                         vertices[U2] = vertices[U4];
192                         vertices[U4] = temp;
193                     }
194                     if (flipY) {
```

```
195                         float temp = vertices[V1];
196                         vertices[V1] = vertices[V3];
197                         vertices[V3] = temp;
198                         temp = vertices[V2];
199                         vertices[V2] = vertices[V4];
200                         vertices[V4] = temp;
201                     }
202                     if (rotations != 0) {
203                         switch (rotations) {
204                             case Cell.ROTATE_90: {
205                                 float tempV = vertices[V1];
206                                 vertices[V1] = vertices[V2];
207                                 vertices[V2] = vertices[V3];
208                                 vertices[V3] = vertices[V4];
209                                 vertices[V4] = tempV;
210 
211                                 float tempU = vertices[U1];
212                                 vertices[U1] = vertices[U2];
213                                 vertices[U2] = vertices[U3];
214                                 vertices[U3] = vertices[U4];
215                                 vertices[U4] = tempU;
216                                 break;
217                             }
218                             case Cell.ROTATE_180: {
219                                 float tempU = vertices[U1];
220                                 vertices[U1] = vertices[U3];
221                                 vertices[U3] = tempU;
222                                 tempU = vertices[U2];
223                                 vertices[U2] = vertices[U4];
224                                 vertices[U4] = tempU;
225                                 float tempV = vertices[V1];
226                                 vertices[V1] = vertices[V3];
227                                 vertices[V3] = tempV;
228                                 tempV = vertices[V2];
229                                 vertices[V2] = vertices[V4];
230                                 vertices[V4] = tempV;
231                                 break;
232                             }
233                             case Cell.ROTATE_270: {
234                                 float tempV = vertices[V1];
235                                 vertices[V1] = vertices[V4];
236                                 vertices[V4] = vertices[V3];
237                                 vertices[V3] = vertices[V2];
238                                 vertices[V2] = tempV;
239 
240                                 float tempU = vertices[U1];
241                                 vertices[U1] = vertices[U4];
242                                 vertices[U4] = vertices[U3];
```

```
243                                 vertices[U3] = vertices[U2];
244                                 vertices[U2] = tempU;
245                                 break;
246                             }
247                         }
248                     }
249 
250                     final float normHeight = (y - viewBounds.y) /
                            viewBounds.height;
251                     getBatch().getShader().setUniformf("tile_depth",
                            normHeight);
252 
253                     batch.draw(region.getTexture(), vertices, 0,
                            NUM_VERTICES);
254                     batch.flush();
255                 }
256             }
257         }
258     }
259 
260     private static ShaderProgram createShader() {
261         final String vertexShader = "attribute␣vec4␣" + ShaderProgram.
                POSITION_ATTRIBUTE + ";\n" //
262                 + "attribute␣vec4␣" + ShaderProgram.COLOR_ATTRIBUTE + "
                    ;\n" //
263                 + "attribute␣vec2␣" + ShaderProgram.TEXCOORD_ATTRIBUTE +
                     "0;\n" //
264                 + "uniform␣mat4␣u_projTrans;\n" //
265                 + "varying␣vec4␣v_color;\n" //
266                 + "varying␣vec2␣v_texCoords;\n" //
267                 + "\n" //
268                 + "void␣main()\n" //
269                 + "{\n" //
270                 + "␣␣␣v_color␣=␣" + ShaderProgram.COLOR_ATTRIBUTE + ";\n
                    " //
271                 + "␣␣␣v_color.a␣=␣v_color.a␣*␣(255.0/254.0);\n" //
272                 + "␣␣␣v_texCoords␣=␣" + ShaderProgram.TEXCOORD_ATTRIBUTE
                     + "0;\n" //
273                 + "␣␣␣gl_Position␣=␣␣u_projTrans␣*␣" + ShaderProgram.
                    POSITION_ATTRIBUTE + ";\n" //
274                 + "}\n";
275         final String fragmentShader = "#ifdef␣GL_ES\n" //
276                 + "#define␣LOWP␣lowp\n" //
277                 + "precision␣mediump␣float;\n" //
278                 + "#else\n" //
279                 + "#define␣LOWP␣\n" //
280                 + "#endif\n" //
281                 + "varying␣LOWP␣vec4␣v_color;\n" //
```

```
282                 + "varying␣vec2␣v_texCoords;\n" //
283                 + "uniform␣sampler2D␣u_texture;\n" //
284                 + "uniform␣float␣tile_depth;\n" //
285                 + "uniform␣float␣alpha_threshold;\n"
286                 + "void␣main()\n" //
287                 + "{\n" //
288                 + "␣␣vec4␣tex␣=␣texture2D(u_texture,␣v_texCoords);\n" //
289                 + "␣␣if(tex.a␣<␣alpha_threshold)\n" //
290                 + "␣␣␣␣␣␣discard;\n"//
291                 + "\n" //
292                 + "␣␣gl_FragColor␣=␣v_color␣*␣tex;\n"
293                 + "␣␣gl_FragDepth␣=␣tile_depth;\n" //
294                 + "}";
295
296         final ShaderProgram shader = new ShaderProgram(vertexShader,
                fragmentShader);
297         if (!shader.isCompiled())
298             throw new IllegalArgumentException("Error␣compiling␣shader:␣
                    " + shader.getLog());
299
300         shader.bind();
301         shader.setUniformf("alpha_threshold", 0.5f);
302
303         return shader;
304     }
305 }
```

SpikeGenerator.java

```
1  package com.dar.freshmaze.level.tilemap;
2
3  import com.badlogic.gdx.math.MathUtils;
4  import com.badlogic.gdx.math.Rectangle;
5  import com.badlogic.gdx.math.Vector2;
6  import com.badlogic.gdx.utils.Array;
7  import com.dar.freshmaze.level.tilemap.rooms.LevelRoom;
8
9  public class SpikeGenerator {
10     // TODO: Add more patterns?
11     public Array<Vector2> generateSpikes(LevelRoom room) {
12         final Rectangle bounds = room.getBounds();
13
14         final Array<Vector2> spikes = new Array<>();
15
16         final float random = MathUtils.random();
17
18         if (random < 0.4f) {
19             return spikes;
```

```
20          } else if (random < 0.65f) {
21              generateSpikesVertical(bounds, spikes);
22          } else if (random < 0.9f) {
23              generateSpikesHorizontal(bounds, spikes);
24          } else {
25              generateSpikesCheckerboard(bounds, spikes);
26          }
27
28          return spikes;
29      }
30
31      private void generateSpikesVertical(Rectangle rect, Array<Vector2>
            spikes) {
32          for (int yi = (int)rect.y; yi < rect.y + rect.height; ++yi) {
33              for (int xi = (int)rect.x; xi < rect.x + rect.width; ++xi) {
34                  if (yi % 3 == 0)
35                      spikes.add(new Vector2(xi, yi));
36              }
37          }
38      }
39
40      private void generateSpikesHorizontal(Rectangle rect, Array<Vector2>
             spikes) {
41          for (int yi = (int)rect.y; yi < rect.y + rect.height; ++yi) {
42              for (int xi = (int)rect.x; xi < rect.x + rect.width; ++xi) {
43                  if (xi % 3 == 0)
44                      spikes.add(new Vector2(xi, yi));
45              }
46          }
47      }
48
49      private void generateSpikesCheckerboard(Rectangle rect, Array<
            Vector2> spikes) {
50          for (int yi = (int)rect.y; yi < rect.y + rect.height; ++yi) {
51              for (int xi = (int)rect.x; xi < rect.x + rect.width; ++xi) {
52                  if ((xi + yi) % 2 == 0)
53                      spikes.add(new Vector2(xi, yi));
54              }
55          }
56      }
57  }
```

Dungeon.java

```
1  package com.dar.freshmaze.level;
2
3  import com.badlogic.gdx.math.Vector2;
4  import com.badlogic.gdx.utils.Disposable;
```

```
5  import com.dar.freshmaze.entities.Bob;
6  import com.dar.freshmaze.level.tilemap.rooms.LevelRoom;
7  import com.dar.freshmaze.util.RectangleUtil;
8
9  public class Dungeon implements Disposable {
10     private final Level level;
11     private final Bob bob;
12     private LevelRoom currentRoom;
13
14     private int levelIndex = 0;
15     private boolean pendingTransition = false;
16
17     public Dungeon(Level level, Bob bob) {
18         this.level = level;
19         this.bob = bob;
20
21         level.getTilemap().setDungeon(this);
22
23         generateLevel();
24     }
25
26     public Level getLevel() {
27         return level;
28     }
29
30     public Bob getBob() {
31         return bob;
32     }
33
34     public boolean isPendingTransition() {
35         return pendingTransition;
36     }
37
38     public void moveToNextLevel() {
39         levelIndex++;
40
41         if (!isMaxLevel())
42             pendingTransition = true;
43     }
44
45     public void update(float dt) {
46         if (pendingTransition) {
47             generateLevel();
48
49             pendingTransition = false;
50         }
51
52         updateRoom();
```

```
53          level.update(dt);
54      }
55
56      private void updateRoom() {
57          final LevelRoom newRoom = findContainingRoom();
58          if (currentRoom == newRoom)
59              return;
60
61          if (currentRoom != null)
62              currentRoom.onPlayerExit(bob);
63
64          currentRoom = newRoom;
65
66          if (currentRoom != null)
67              currentRoom.onPlayerEnter(bob);
68      }
69
70      private LevelRoom findContainingRoom() {
71          final Vector2 bobWorldPos = new Vector2(bob.getX() + 0.5f * bob.
                getWidth(), bob.getY() + 0.5f * bob.getHeight());
72          final Vector2 bobCellPos = level.getTilemap().vecToCellPosVec(
                bobWorldPos);
73
74          for (LevelRoom room : level.getRooms()) {
75              if (RectangleUtil.containsExclusive(room.getBounds(),
                    bobCellPos))
76                  return room;
77          }
78
79          return null;
80      }
81
82      private void generateLevel() {
83          final LevelRoom startRoom = level.generate(levelIndex);
84          final Vector2 centerCell = startRoom.getBounds().getCenter(new
                Vector2());
85          final Vector2 center = level.getTilemap().cellPosToVec(
                centerCell);
86
87          bob.teleport(center);
88      }
89
90      public boolean isMaxLevel() {
91          return getLevelIndex() == level.getMaxLevel();
92      }
93
94      public int getLevelIndex() {
95          return levelIndex;
```

```
96      }
97
98      @Override
99      public void dispose() {
100         level.dispose();
101     }
102 }
```

EnemyGenerator.java

```
1   package com.dar.freshmaze.level;
2
3   import com.badlogic.gdx.math.MathUtils;
4   import com.badlogic.gdx.math.Rectangle;
5   import com.badlogic.gdx.math.Vector2;
6   import com.badlogic.gdx.physics.box2d.World;
7   import com.badlogic.gdx.scenes.scene2d.Stage;
8   import com.badlogic.gdx.utils.Array;
9   import com.dar.freshmaze.entities.Enemy;
10  import com.dar.freshmaze.entities.Entity;
11  import com.dar.freshmaze.entities.HealthBonus;
12  import com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom;
13  import com.dar.freshmaze.util.RectangleUtil;
14
15  public class EnemyGenerator {
16      private final World physWorld;
17      private final Stage stage;
18      private Dungeon dungeon;
19
20      public EnemyGenerator(World physWorld, Stage stage) {
21          this.physWorld = physWorld;
22          this.stage = stage;
23      }
24
25      public void setDungeon(Dungeon dungeon) {
26          this.dungeon = dungeon;
27      }
28
29      public Result generate(BattleLevelRoom room) {
30          final Array<Enemy> enemies = new Array<>();
31          final Array<Entity> otherEntities = new Array<>();
32
33          if(MathUtils.randomBoolean())
34              enemies.add(createEnemy(room));
35          if(MathUtils.randomBoolean() && dungeon.getLevelIndex() >= 1)
36              enemies.add(createEnemy(room));
37          if(MathUtils.randomBoolean() && MathUtils.randomBoolean() &&
               dungeon.getLevelIndex() >= 2)
```

```
38              enemies.add(createEnemy(room));
39          if(MathUtils.random(3) >= dungeon.getLevelIndex())
40              otherEntities.add(createHealthBouns(room));
41
42          enemies.add(createEnemy(room));
43          enemies.forEach(stage::addActor);
44          otherEntities.forEach(stage::addActor);
45
46          return new Result(enemies, otherEntities);
47      }
48
49      private HealthBonus createHealthBouns(BattleLevelRoom room) {
50          return new HealthBonus(physWorld, room, getSpawnPos(
                RectangleUtil.shrink(room.getBounds(), new Vector2(1.0f, 1.0
                f))));
51      }
52
53      private Enemy createEnemy(BattleLevelRoom room) {
54          return new Enemy(physWorld, room, getSpawnPos(room.getBounds()))
                ;
55      }
56
57      private static Vector2 getSpawnPos(Rectangle rect) {
58          return RectangleUtil.getRandomPoint(rect);
59      }
60
61      public static class Result {
62          public final Array<Enemy> enemies;
63          public final Array<Entity> otherEntities;
64
65          public Result(Array<Enemy> enemies, Array<Entity> otherEntities)
                {
66              this.enemies = enemies;
67              this.otherEntities = otherEntities;
68          }
69      }
70  }
```

Level.java

```
1  package com.dar.freshmaze.level;
2
3  import com.badlogic.gdx.graphics.Camera;
4  import com.badlogic.gdx.graphics.Color;
5  import com.badlogic.gdx.graphics.OrthographicCamera;
6  import com.badlogic.gdx.graphics.glutils.ShapeRenderer;
7  import com.badlogic.gdx.math.MathUtils;
8  import com.badlogic.gdx.math.Matrix4;
```

```
 9 import com.badlogic.gdx.math.Rectangle;
10 import com.badlogic.gdx.math.Vector2;
11 import com.badlogic.gdx.physics.box2d.World;
12 import com.badlogic.gdx.scenes.scene2d.Stage;
13 import com.badlogic.gdx.utils.Disposable;
14 import com.dar.freshmaze.level.bitmap.LevelBitmap;
15 import com.dar.freshmaze.level.graph.*;
16 import com.dar.freshmaze.level.tilemap.LevelTilemap;
17 import com.dar.freshmaze.level.tilemap.SortedIsometricTiledMapRenderer;
18 import com.dar.freshmaze.level.tilemap.SpikeGenerator;
19 import com.dar.freshmaze.level.tilemap.rooms.LevelRoom;
20 import com.dar.freshmaze.util.IsometricUtil;
21
22 import java.util.*;
23 import java.util.stream.Collectors;
24 import java.util.stream.Stream;
25
26 public class Level implements Disposable {
27     private final LevelNodeGenerator nodeGenerator;
28     private final EnemyGenerator enemyGenerator;
29     private final SpikeGenerator spikeGenerator;
30     private final LevelTilemap tilemap;
31     private SortedIsometricTiledMapRenderer tilemapRenderer;
32
33     private final ShapeRenderer shape;
34     private List<Color> debugLeafColors;
35     private Matrix4 debugRenderMatrix;
36
37     private final static int MAX_LEVEL = 3;
38
39     private final Vector2[] levelSizes = {
40         new Vector2(38, 38),
41         new Vector2(50, 50),
42         new Vector2(64, 64)
43     };
44
45     private final LevelNodeGenerationRules[] generationRules = {
46             new LevelNodeGenerationRules(6, 12, 20, 0.75f, 2),
47             new LevelNodeGenerationRules(8, 12, 25, 0.75f, 2),
48             new LevelNodeGenerationRules(8, 14, 30, 0.75f, 2)
49     };
50
51     public Level(World physWorld, Stage stage) {
52         nodeGenerator = new LevelNodeGenerator();
53         tilemap = new LevelTilemap(physWorld, "level/tiles/tiles.png",
               1.0f, 128);
54         enemyGenerator = new EnemyGenerator(physWorld, stage);
55         spikeGenerator = new SpikeGenerator();
```

```
56
57          shape = new ShapeRenderer();
58      }
59
60      public LevelRoom generate(int levelIndex) {
61          enemyGenerator.setDungeon(getTilemap().getDungeon());
62
63          generateNodes(levelIndex);
64
65          final LevelBitmap levelBitmap = new LevelBitmap();
66          levelBitmap.generate(nodeGenerator);
67
68          tilemap.generate(levelBitmap);
69          tilemapRenderer = new SortedIsometricTiledMapRenderer(tilemap.
                getTilemap(), tilemap.getTileSize() / tilemap.
                getTextureTileSize());
70          debugLeafColors = Stream
71                  .generate(() -> new Color(MathUtils.random(0, 0xFFFFFF))
                        )
72                  .limit(nodeGenerator.getRooms().size())
73                  .collect(Collectors.toList());
74          debugRenderMatrix = new Matrix4()
75                  .idt()
76                  .scl(tilemap.getTileSize(), tilemap.getTileSize(), 1.0f)
77                  .mul(IsometricUtil.ISO_TRANSFORMATION_MATRIX);
78
79          nodeGenerator.getRooms().forEach(room -> room.setLevel(this));
80
81          return nodeGenerator.getSpawnRoom();
82      }
83
84      public int getMaxLevel() {
85          return MAX_LEVEL;
86      }
87
88      private void generateNodes(int levelIndex) {
89          nodeGenerator.generate(
90                  levelSizes[levelIndex],
91                  2,
92                  generationRules[levelIndex],
93                  enemyGenerator,
94                  spikeGenerator
95          );
96      }
97
98      public List<LevelRoom> getRooms() {
99          return Collections.unmodifiableList(nodeGenerator.getRooms());
100     }
```

```
101
102     public LevelTilemap getTilemap() {
103         return tilemap;
104     }
105
106     public SortedIsometricTiledMapRenderer getTilemapRenderer() {
107         return tilemapRenderer;
108     }
109
110     public void update(float dt) {
111         nodeGenerator.getRooms().forEach(room -> room.act(dt));
112     }
113
114     public void render(OrthographicCamera camera, float dt, int[] layers
        , boolean writeDepth) {
115         if (tilemapRenderer == null)
116             return;
117
118         tilemapRenderer.setWriteDepth(writeDepth);
119         tilemapRenderer.setView(camera);
120         tilemapRenderer.render(layers);
121     }
122
123     public void debugRender(Camera camera, float dt, int kind) {
124         shape.setTransformMatrix(debugRenderMatrix);
125         shape.setProjectionMatrix(camera.combined);
126
127         if ((kind & DebugRender.LEAVES) == DebugRender.LEAVES)
128             debugRenderLeaves();
129         if ((kind & DebugRender.HALLS) == DebugRender.HALLS)
130             debugRenderHalls();
131         if ((kind & DebugRender.GRAPH) == DebugRender.GRAPH)
132             debugRenderGraph();
133         if ((kind & DebugRender.ROOMS) == DebugRender.ROOMS)
134             debugRenderRooms();
135         if ((kind & DebugRender.GRID) == DebugRender.GRID)
136             debugRenderGrid();
137     }
138
139     private void debugRenderGrid() {
140         final Vector2 levelSize = nodeGenerator.getLevelSize();
141
142         shape.begin(ShapeRenderer.ShapeType.Filled);
143         shape.setColor(Color.BLACK);
144
145         for (int xi = 0; xi <= levelSize.x; ++xi)
146             shape.rectLine(xi, 0.0f, xi, levelSize.y, 0.1f);
147
```

```
148         for (int yi = 0; yi <= levelSize.y; ++yi)
149             shape.rectLine(0.0f, yi, levelSize.x, yi, 0.1f);
150 
151         shape.end();
152     }
153 
154     private void debugRenderRooms() {
155         int index = 0;
156 
157         shape.begin(ShapeRenderer.ShapeType.Filled);
158         for (LevelRoom room : nodeGenerator.getRooms()) {
159             shape.setColor(debugLeafColors.get(index++).cpy().mul(1.0f,
                    1.0f, 1.0f, 0.1f));
160             shape.rect(room.getBounds().x, room.getBounds().y, room.
                    getBounds().width, room.getBounds().height);
161         }
162         shape.end();
163     }
164 
165     private void debugRenderLeaves() {
166         int index = 0;
167 
168         shape.begin(ShapeRenderer.ShapeType.Filled);
169         for (LevelNode leaf : nodeGenerator.getLeaves()) {
170             shape.setColor(debugLeafColors.get(index++));
171             shape.rect(leaf.getBounds().x, leaf.getBounds().y, leaf.
                    getBounds().width, leaf.getBounds().height);
172         }
173         shape.end();
174     }
175 
176     private void debugRenderGraph() {
177         shape.begin(ShapeRenderer.ShapeType.Filled);
178 
179         for (Map.Entry<LevelNode, HashMap<LevelNode, LevelGraph.Edge>>
                entry : nodeGenerator.getGraph().entrySet()) {
180             final LevelNode firstNode = entry.getKey();
181 
182             for (Map.Entry<LevelNode, LevelGraph.Edge> connection :
                    entry.getValue().entrySet()) {
183                 final LevelNode secondNode = connection.getKey();
184 
185                 shape.setColor(Color.BLACK);
186                 shape.line(firstNode.getRoomBounds().getCenter(new
                        Vector2()), secondNode.getRoomBounds().getCenter(new
                         Vector2()));
187             }
188         }
```

```
189
190             shape.end();
191         }
192
193         private void debugRenderHalls() {
194             shape.begin(ShapeRenderer.ShapeType.Filled);
195             for (Rectangle hall : nodeGenerator.getHalls()) {
196                 shape.setColor(Color.ORANGE);
197                 shape.rect(hall.x, hall.y, hall.width, hall.height);
198             }
199             shape.end();
200         }
201
202         public static class DebugRender {
203             public static final int NONE = 0;
204             public static final int GRAPH = 1 << 0;
205             public static final int LEAVES = 1 << 1;
206             public static final int HALLS = 1 << 2;
207             public static final int ROOMS = 1 << 3;
208             public static final int GRID = 1 << 4;
209         }
210         @Override
211         public void dispose() {
212             if (tilemapRenderer != null)
213                 tilemapRenderer.dispose();
214         }
215 }
```

RectangleUtil.java

```
1  package com.dar.freshmaze.util;
2
3  import com.badlogic.gdx.math.MathUtils;
4  import com.badlogic.gdx.math.Rectangle;
5  import com.badlogic.gdx.math.Vector2;
6
7  public class RectangleUtil {
8      public static Vector2 getRandomPoint(Rectangle rect) {
9          return new Vector2(
10                 (int)rect.x + MathUtils.random((int)rect.width - 1),
11                 (int)rect.y + MathUtils.random((int)rect.height - 1)
12         );
13     }
14
15     /**
16      * Expands rectangle by delta in all directions
17      *
18      * @param rect rectangle to expand
```

```
19          * @param delta size by which to expand the rectangle
20          * @return rect
21          */
22         public static Rectangle expand(Rectangle rect, Vector2 delta) {
23             return new Rectangle(
24                     rect.x - delta.x,
25                     rect.y - delta.y,
26                     rect.width + 2.0f * delta.x,
27                     rect.height + 2.0f * delta.y
28             );
29         }
30
31         /**
32          * Shrinks rectangle by delta in all directions
33          *
34          * @param rect rectangle to shrink
35          * @param delta size by which to shrink the rectangle
36          * @return rect
37          */
38         public static Rectangle shrink(Rectangle rect, Vector2 delta) {
39             return expand(rect, delta.scl(-1.0f));
40         }
41
42         /**
43          * Normalizes a rectangle, in other words,
44          * makes sure it has strictly positive size
45          * and modifies its position accordingly
46          *
47          * @param rect rectangle to normalize
48          * @return rect
49          */
50         public static Rectangle normalize(Rectangle rect) {
51             if (rect.width < 0) {
52                 rect.width = -rect.width;
53                 rect.x -= rect.width;
54             }
55             if (rect.height < 0) {
56                 rect.height = -rect.height;
57                 rect.y -= rect.height;
58             }
59
60             return rect;
61         }
62
63         public static boolean containsExclusive(Rectangle rect, Vector2
               point) {
64             return rect.x <= point.x && rect.x + rect.width > point.x &&
65                     rect.y <= point.y && rect.y + rect.height > point.y;
```

```
66      }
67  }
```

IsometricUtil.java

```
1   package com.dar.freshmaze.util;
2
3   import com.badlogic.gdx.math.Matrix4;
4   import com.badlogic.gdx.math.Vector2;
5
6   /**
7    * Class with various helper methods for
8    * woring with isometric coordinate system
9    */
10  public class IsometricUtil {
11      /** Matrix that can be used to transform coordinates to isometric
            for rendering (ot other purposes).
12       * Doesn't need to be used if the sprites are already drawn as
             isometric.
13       *
14       * Can be used like:
15       * <pre>
16       * {@code game.batch.setTransformMatrix(IsometricUtil.
             ISO_TRANSFORMATION_MATRIX); }
17       * </pre>
18       * or
19       * <pre>
20       * {@code physDebugRenderer.render(physWorld, camera.combined.mul(
             IsometricUtil.ISO_TRANSFORMATION_MATRIX)); }
21       * </pre>
22       *
23       */
24      public static final Matrix4 ISO_TRANSFORMATION_MATRIX;
25
26      static {
27          ISO_TRANSFORMATION_MATRIX = new Matrix4()
28              .idt()
29              .scale((float)(Math.sqrt(2.0) / 2.0), (float)(Math.sqrt(2.0)
                    / 4.0), 1.0f)
30              .rotate(0.0f, 0.0f, 1.0f, -45);
31      }
32
33      /**
34       * Converts vector from isometric to cartesian coordinate system.
35       *
36       * The original formula is the following:
37       * <pre>
38       * {@code
```

```
39      * return new Vector2(
40      *      0.5f * (2.0f * vec.y + vec.x),
41      *      0.5f * (2.0f * vec.y - vec.x)
42      * );
43      * }
44      * </pre>
45      * But we need to rotate it to match our tilemap. The result is:
46      * <pre>
47      * {@code
48      *  return new Vector2(
49      *      0.5f * (-2.0f * vec.y + vec.x),
50      *      0.5f * (2.0f * vec.y + vec.x)
51      *  );
52      *  }
53      *  </pre>
54      *
55      * @param vec vector in isometric system
56      * @return vector in cartesian system
57      */
58     public static Vector2 isoToCart(Vector2 vec) {
59         return new Vector2(
60                 0.5f * (-2.0f * vec.y + vec.x),
61                 0.5f * (2.0f * vec.y + vec.x)
62         );
63     }
64
65     /**
66      * Converts vector from cartesian to isometric coordinate system.
67      *
68      * The original formula is the following:
69      * <pre>
70      * {@code
71      * return new Vector2(
72      *      vec.x - vec.y,
73      *      0.5f * (vec.x + vec.y)
74      * );
75      * }
76      * </pre>
77      * But we need to rotate it to match our tilemap. The result is:
78      * <pre>
79      * {@code
80      *  return new Vector2(
81      *      vec.x + vec.y,
82      *      0.5f * (vec.x - vec.y)
83      *  );
84      *  }
85      *  </pre>
86      *
```

```
87          *  NOTE: I have no idea where multiplication of both
88          *  x and y by 0.5 comes from, it makes no sense. Fix it if possible
89          *
90          * @param vec vector in cartesian system
91          * @return vector in isometric system
92          */
93         public static Vector2 cartToIso(Vector2 vec) {
94             return new Vector2(
95                     0.5f * (vec.x + vec.y),
96                     0.25f * (vec.y - vec.x)
97             );
98         }
99     }
```

TimeUtil.java

```
1  package com.dar.freshmaze.util;
2
3  import com.badlogic.gdx.utils.TimeUtils;
4
5  public class TimeUtil {
6      private static long startTime;
7
8      public static void init() {
9          startTime = TimeUtils.millis();
10     }
11
12     public static long time() {
13         return TimeUtils.timeSinceMillis(startTime);
14     }
15
16     public static float timef() {
17         return time() / 1000.0f;
18     }
19 }
```

Bob.java

```
1  package com.dar.freshmaze.entities;
2
3  import com.badlogic.gdx.Gdx;
4  import com.badlogic.gdx.Input;
5  import com.badlogic.gdx.audio.Sound;
6  import com.badlogic.gdx.graphics.Color;
7  import com.badlogic.gdx.graphics.Texture;
8  import com.badlogic.gdx.math.Vector2;
9  import com.badlogic.gdx.physics.box2d.*;
10 import com.badlogic.gdx.scenes.scene2d.*;
11 import com.badlogic.gdx.graphics.g2d.*;
```

```
12 import com.badlogic.gdx.scenes.scene2d.actions.ColorAction;
13 import com.badlogic.gdx.scenes.scene2d.actions.SequenceAction;
14 import com.badlogic.gdx.utils.Array;
15 import com.dar.freshmaze.level.Level;
16 import com.dar.freshmaze.level.tilemap.tiles.DynamicInteractableTile;
17 import com.dar.freshmaze.level.tilemap.tiles.SpikesTile;
18 import com.dar.freshmaze.util.IsometricUtil;
19
20
21 /**
22  * Class that represents Bob The Player.
23  */
24 public class Bob extends Entity {
25     private final Texture attackTexture = new Texture(Gdx.files.internal
          ("iso_circle_marker.png"));
26     public static final float MOVEMENT_SPEED = 4.0f;
27     public static final float deltaPx = 1.0f;
28     public static final float deltaPy = 1.0f;
29     public boolean movingRight = false;
30     public boolean movingLeft = false;
31     public boolean movingUp = false;
32     public boolean movingDown = false;
33
34     private boolean isAttacking = false;
35     private boolean isInteracting = false;
36     private final static float multiplier = 1.6f;
37     private Level level;
38     private int health;
39
40     private final static float maxAttackSpeed = 1.5f;
41     private final static float attackDisplayMult = 0.1f;
42
43     private float attackSpeed = 0.5f;
44     private float attackTimeLeft;
45
46     private final Array<Object> closeObjects = new Array<>();
47
48     public Bob(World physWorld, Level level, Vector2 spawnPos) {
49         super(physWorld, createSprite(), createBody(physWorld), new
              Vector2(0.5f, -0.5f), SpriteKind.Isometric, spawnPos);
50
51         this.level = level;
52         health = getMaxHealth();
53
54         addListener(new InputListener() {
55             @Override
56             public boolean keyDown(InputEvent event, int keycode) {
57                 if (keycode == Input.Keys.D)
```

```
58                      movingRight = true;
59                  if (keycode == Input.Keys.A)
60                      movingLeft = true;
61                  if (keycode == Input.Keys.W)
62                      movingUp = true;
63                  if (keycode == Input.Keys.S)
64                      movingDown = true;
65                  if (keycode == Input.Keys.SPACE)
66                      isAttacking = true;
67                  if (keycode == Input.Keys.E)
68                      isInteracting = true;
69
70                  return true;
71              }
72
73              @Override
74              public boolean keyUp(InputEvent event, int keycode) {
75                  if (keycode == Input.Keys.D)
76                      movingRight = false;
77                  if (keycode == Input.Keys.A)
78                      movingLeft = false;
79                  if (keycode == Input.Keys.W)
80                      movingUp = false;
81                  if (keycode == Input.Keys.S)
82                      movingDown = false;
83                  if (keycode == Input.Keys.SPACE)
84                      isAttacking = false;
85                  if (keycode == Input.Keys.E)
86                      isInteracting = false;
87
88                  return true;
89              }
90          });
91      }
92
93      /**
94       * Returns the attack time left
95       * @return attack time left
96       */
97      public float getAttackTimeLeft() {
98          return attackTimeLeft;
99      }
100
101     public float getTimePerAttack() {
102         return 1.0f / attackSpeed;
103     }
104
105     /**
```

```
106         * Damage the bob
107         * @param damage the delta for the health.
108         */
109        public void damage(int damage) {
110            setHealth(getHealth() - damage);
111
112            onDamage();
113        }
114
115        /**
116         * Heal the bob
117         * @param amount the delta for the heal
118         */
119        public void heal(int amount) {
120            setHealth(Math.min(getHealth() + amount, getMaxHealth()));
121
122            onHeal();
123        }
124
125        /**
126         * Increase the attack speed
127         * @param amount the amount to increase the attack speed
128         */
129        public void increaseAttackSpeed(float amount) {
130            attackSpeed = Math.min(attackSpeed + amount, maxAttackSpeed);
131
132            onIncreaseAttackSpeed();
133        }
134
135        @Override
136        public void draw(Batch batch, float alpha) {
137            if (attackTimeLeft >= (1.0f - attackDisplayMult) *
                    getTimePerAttack()) {
138                final Vector2 pos = IsometricUtil.cartToIso(new Vector2(getX
                        () - 0.5f, getY() - 0.5f));
139
140                setShaderSortHeight(batch, 1.0f);
141                batch.setColor(Color.RED);
142                batch.draw(attackTexture, pos.x, pos.y - 0.5f, 2.0f, 2.0f);
143                batch.flush();
144            }
145
146            getSprite().setColor(getColor());
147
148            super.draw(batch, alpha);
149        }
150
151        @Override
```

```
152     public void act(float delta) {
153         super.act(delta);
154         if (isDestroyed())
155             return;
156 
157         attackTimeLeft -= delta;
158 
159         processActions();
160 
161         float dx = 0;
162         float dy = 0;
163         if (movingRight)
164             dx = deltaPx;
165         if (movingLeft)
166             dx = -deltaPx;
167         if (movingUp)
168             dy = deltaPy;
169         if (movingDown)
170             dy = -deltaPy;
171 
172         getBody().setLinearVelocity(IsometricUtil.isoToCart(new Vector2(
                dx, dy).nor().scl(MOVEMENT_SPEED)));
173     }
174 
175     public void addObjectInRadius(Object userData) {
176         if (userData == null)
177             return;
178 
179         if (!processContact(userData))
180             closeObjects.add(userData);
181     }
182 
183     public void removeObjectInRadius(Object userData) {
184         if (userData == null)
185             return;
186 
187         closeObjects.removeValue(userData, true);
188     }
189 
190     private void processActions() {
191         boolean attacked = false;
192         if (isAttacking) {
193             if (attackTimeLeft <= 0) {
194                 attacked = true;
195                 attackTimeLeft = getTimePerAttack();
196                 Sound sound = Gdx.audio.newSound(Gdx.files.internal("
                        attack.mp3"));
197                 sound.setLooping(sound.play(1.0f), false);
```

```
198             }
199         }
200 
201         for (Object obj : closeObjects) {
202             if (obj instanceof Enemy) {
203                 if (attacked)
204                     ((Enemy) obj).kill();
205             } else if (obj instanceof DynamicInteractableTile) {
206                 if (isInteracting)
207                     ((DynamicInteractableTile) obj).interact(this);
208             } else if(obj instanceof HealthBonus) {
209                 ((HealthBonus) obj).interact(this);
210             }
211         }
212     }
213 
214     private boolean processContact(Object obj) {
215         if (obj instanceof Enemy) {
216             damage(7);
217         } else if (obj instanceof SpikesTile) {
218             ((SpikesTile)obj).onTouch(this);
219 
220             return true;
221         }
222 
223         return false;
224     }
225 
226     private void onDamage() {
227         ColorAction ca = new ColorAction();
228         ca.setEndColor(Color.RED);
229         ca.setDuration(0.4f);
230         ColorAction ca1 = new ColorAction();
231         ca1.setEndColor(Color.WHITE);
232         ca1.setDuration(0.4f);
233         SequenceAction sequenceAction = new SequenceAction();
234         sequenceAction.addAction(ca);
235         sequenceAction.addAction(ca1);
236         addAction(sequenceAction);
237 
238         addAction(ca);
239     }
240 
241     private void onHeal() {
242         ColorAction ca = new ColorAction();
243         ca.setEndColor(Color.GREEN);
244         ca.setDuration(0.4f);
245         ColorAction ca1 = new ColorAction();
```

```
246         ca1.setEndColor(Color.WHITE);
247         ca1.setDuration(0.4f);
248         SequenceAction sequenceAction = new SequenceAction();
249         sequenceAction.addAction(ca);
250         sequenceAction.addAction(ca1);
251         addAction(sequenceAction);
252 
253         addAction(ca);
254     }
255 
256     private void onIncreaseAttackSpeed() {
257         ColorAction ca = new ColorAction();
258         ca.setEndColor(Color.TEAL);
259         ca.setDuration(0.6f);
260         ColorAction ca1 = new ColorAction();
261         ca1.setEndColor(Color.WHITE);
262         ca1.setDuration(0.6f);
263         SequenceAction sequenceAction = new SequenceAction();
264         sequenceAction.addAction(ca);
265         sequenceAction.addAction(ca1);
266         addAction(sequenceAction);
267 
268         addAction(ca);
269     }
270 
271     private static Sprite createSprite() {
272         final Sprite sprite = new Sprite(new Texture(Gdx.files.internal(
                "player.png")));
273         sprite.setSize(1.0f, 1.0f);
274 
275         return sprite;
276     }
277 
278     private static Body createBody(World physWorld) {
279         final CircleShape circle = new CircleShape();
280         circle.setPosition(new Vector2(0.5f, 0.5f));
281         circle.setRadius(0.3f);
282 
283         final BodyDef bd = new BodyDef();
284         bd.type = BodyDef.BodyType.DynamicBody;
285         bd.fixedRotation = true;
286         bd.linearDamping = 1f;
287         bd.angularDamping = 1f;
288 
289         final CircleShape circleSensor = new CircleShape();
290         circleSensor.setPosition(new Vector2(0.5f, 0.5f));
291         circleSensor.setRadius(0.5f * multiplier);
292 
```

```
293         final FixtureDef fdef = new FixtureDef();
294         fdef.shape = circleSensor;
295         fdef.isSensor = true;
296 
297         final Body body = physWorld.createBody(bd);
298         body.createFixture(circle, 1);
299         body.createFixture(fdef);
300 
301         return body;
302     }
303     public void setHealth(int health) {
304         this.health = health;
305     }
306     public int getHealth() {
307         return health;
308     }
309 
310     public int getMaxHealth() {
311         return 100;
312     }
313 }
```

Enemy.java

```
1  package com.dar.freshmaze.entities;
2  
3  import com.badlogic.gdx.Gdx;
4  import com.badlogic.gdx.graphics.Texture;
5  import com.badlogic.gdx.math.MathUtils;
6  import com.badlogic.gdx.math.Vector2;
7  import com.badlogic.gdx.physics.box2d.Body;
8  import com.badlogic.gdx.physics.box2d.BodyDef;
9  import com.badlogic.gdx.physics.box2d.CircleShape;
10 import com.badlogic.gdx.physics.box2d.World;
11 import com.badlogic.gdx.graphics.g2d.*;
12 import com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom;
13 import com.badlogic.gdx.physics.box2d.FixtureDef;
14 
15 public class Enemy extends Entity {
16     public final float movementSpeed;
17     public final float deltaPx;
18     public final float deltaPy;
19     private final int boxSize;
20     private int boxIndex = 0;
21     private boolean boxForward = true;
22 
23     private final BattleLevelRoom room;
24 
```

```
25      public Enemy(World physWorld, BattleLevelRoom room, Vector2 spawnPos
            ) {
26          super(physWorld, createSprite(), createBody(physWorld), new
                Vector2(0.5f, -0.5f), SpriteKind.Isometric, spawnPos);
27
28          this.room = room;
29
30          deltaPx = MathUtils.random(2.0f);
31          deltaPy = MathUtils.random(2.0f);
32          boxSize = MathUtils.random(300) + 50;
33          movementSpeed = (MathUtils.random() + 0.5f) * 2;
34      }
35
36      public void kill() {
37          room.onEnemyDeath(this);
38
39          destroy();
40      }
41
42      @Override
43      public void act(float delta) {
44          super.act(delta);
45          if (isDestroyed())
46              return;
47
48  //        if (++boxIndex == boxSize ) {
49  //            boxForward = !boxForward;
50  //            boxIndex = 0;
51  //        }
52          if(getBody().getPosition().x - deltaPx * movementSpeed < room.
                getBounds().x || getBody().getPosition().y - deltaPy *
                movementSpeed < room.getBounds().y) {
53              boxForward = true;
54  //            boxIndex = 0;
55          } else if(getBody().getPosition().x + deltaPx * movementSpeed >
                room.getBounds().x + room.getBounds().width || getBody().
                getPosition().y + deltaPy * movementSpeed > room.getBounds()
                .y + room.getBounds().height) {
56              boxForward = false;
57  //            boxIndex = 0;
58          }
59
60          if (!boxForward) {
61              getBody().setLinearVelocity(new Vector2(-deltaPx, -deltaPy).
                    scl(movementSpeed));
62          } else {
63              getBody().setLinearVelocity(new Vector2(deltaPx, deltaPy).
                    scl(movementSpeed));
```

```
64          }
65      }
66
67      private static Sprite createSprite() {
68          final Sprite sprite = new Sprite(new Texture(Gdx.files.internal(
                "enemy.png")));
69          sprite.setSize(1.0f, 1.0f);
70
71          return sprite;
72      }
73
74      private static Body createBody(World physWorld) {
75          final CircleShape circle = new CircleShape();
76          circle.setPosition(new Vector2(0.5f, 0.5f));
77          circle.setRadius(0.5f);
78
79          final BodyDef bd = new BodyDef();
80          bd.type = BodyDef.BodyType.DynamicBody;
81          bd.fixedRotation = true;
82          bd.linearDamping = MathUtils.random();
83          bd.angularDamping = MathUtils.random();
84
85          final FixtureDef fdef = new FixtureDef();
86          fdef.shape = circle;
87
88          final Body body = physWorld.createBody(bd);
89          body.createFixture(circle, 1);
90          body.createFixture(fdef);
91
92          return body;
93      }
94  }
```

HealthBonus.java

```
1   package com.dar.freshmaze.entities;
2
3   import com.badlogic.gdx.Gdx;
4   import com.badlogic.gdx.graphics.Texture;
5   import com.badlogic.gdx.graphics.glutils.ShaderProgram;
6   import com.badlogic.gdx.math.Vector2;
7   import com.badlogic.gdx.physics.box2d.Body;
8   import com.badlogic.gdx.physics.box2d.BodyDef;
9   import com.badlogic.gdx.physics.box2d.CircleShape;
10  import com.badlogic.gdx.physics.box2d.World;
11  import com.badlogic.gdx.graphics.g2d.*;
12  import com.dar.freshmaze.graphics.DepthSortedStage;
13  import com.dar.freshmaze.level.tilemap.rooms.BattleLevelRoom;
```

```
14 import com.badlogic.gdx.physics.box2d.FixtureDef;
15 import com.dar.freshmaze.util.TimeUtil;
16
17 public class HealthBonus extends Entity {
18     private final static ShaderProgram shader = createShader();
19     private final BattleLevelRoom room;
20
21
22     public HealthBonus(World physWorld, BattleLevelRoom room, Vector2
           spawnPos) {
23         super(physWorld, createSprite(), createBody(physWorld), new
               Vector2(0.25f, 0.25f), SpriteKind.Isometric, spawnPos);
24
25         this.room = room;
26     }
27
28     public void interact(Bob bob) {
29         bob.heal(10);
30
31         remove();
32     }
33
34     @Override
35     public void draw(Batch batch, float alpha) {
36         final ShaderProgram oldShader = batch.getShader();
37         batch.setShader(shader);
38         ((DepthSortedStage)getStage()).shaderSetVerticalViewBounds();
39         shader.setUniformf("time", 2.0f * TimeUtil.timef());
40
41         super.draw(batch, alpha);
42
43         batch.setShader(oldShader);
44     }
45
46     private static Sprite createSprite() {
47         final Sprite sprite = new Sprite(new Texture(Gdx.files.internal(
               "heart.png")));
48         sprite.setSize(0.5f, 0.5f);
49
50         return sprite;
51     }
52
53     private static Body createBody(World physWorld) {
54         final CircleShape circle = new CircleShape();
55         circle.setPosition(new Vector2(0.5f, 0.5f));
56         circle.setRadius(0.5f);
57
58         final BodyDef bd = new BodyDef();
```

```
59          bd.type = BodyDef.BodyType.StaticBody;
60          bd.fixedRotation = true;
61          bd.gravityScale = 0;
62
63          final FixtureDef fdef = new FixtureDef();
64          fdef.shape = circle;
65
66          final Body body = physWorld.createBody(bd);
67          body.createFixture(circle, 1);
68          body.createFixture(fdef);
69
70          return body;
71      }
72
73      private static ShaderProgram createShader() {
74          final String vertexShader = "attribute␣vec4␣" + ShaderProgram.
                POSITION_ATTRIBUTE + ";\n" //
75                  + "attribute␣vec4␣" + ShaderProgram.COLOR_ATTRIBUTE + "
                        ;\n" //
76                  + "attribute␣vec2␣" + ShaderProgram.TEXCOORD_ATTRIBUTE +
                          "0;\n" //
77                  + "uniform␣mat4␣u_projTrans;\n" //
78                  + "varying␣vec4␣v_color;\n" //
79                  + "varying␣vec2␣v_texCoords;\n" //
80                  + "\n" //
81                  + "void␣main()\n" //
82                  + "{\n" //
83                  + "␣␣␣␣v_color␣=␣" + ShaderProgram.COLOR_ATTRIBUTE + ";\n
                        " //
84                  + "␣␣␣␣v_color.a␣=␣v_color.a␣*␣(255.0/254.0);\n" //
85                  + "␣␣␣␣v_texCoords␣=␣" + ShaderProgram.TEXCOORD_ATTRIBUTE
                          + "0;\n" //
86                  + "␣␣␣␣gl_Position␣=␣␣u_projTrans␣*␣" + ShaderProgram.
                        POSITION_ATTRIBUTE + ";\n" //
87                  + "}\n";
88          final String fragmentShader = "#ifdef␣GL_ES\n" //
89                  + "#define␣LOWP␣lowp\n" //
90                  + "precision␣mediump␣float;\n" //
91                  + "#else\n" //
92                  + "#define␣LOWP␣\n" //
93                  + "#endif\n" //
94                  + "varying␣LOWP␣vec4␣v_color;\n" //
95                  + "varying␣vec2␣v_texCoords;\n" //
96                  + "uniform␣sampler2D␣u_texture;\n" //
97                  + "uniform␣float␣height;\n" //
98                  + "uniform␣vec2␣bounds_vert;\n" //
99                  + "uniform␣float␣alpha_threshold;\n" //
100                 + "uniform␣float␣time;\n" //
```

```
                + "void␣main()\n" //
                + "{\n" //
                + "␣␣vec2␣uv␣=␣v_texCoords;\n" //
                + "␣␣uv.x␣=␣(uv.x␣-␣0.5)␣*␣2.0;\n" //
                + "␣␣uv.x␣/=␣sin(time);\n" //
                + "␣␣uv.x␣=␣uv.x␣/␣2.0␣+␣0.5;\n" //
                + "␣␣\n" //
                + "␣␣vec4␣tex␣=␣texture2D(u_texture,␣uv);\n" //
                + "␣␣tex.a␣*=␣step(uv.x,␣1.0)␣*␣step(0.0,␣uv.x);\n" //
                + "␣␣\n" //
                + "␣␣if(tex.a␣<␣alpha_threshold)\n" //
                + "␣␣␣␣␣␣discard;\n"//
                + "\n" //
                + "␣␣gl_FragColor␣=␣v_color␣*␣tex;\n"
                + "␣␣gl_FragDepth␣=␣(height␣-␣bounds_vert.x)␣/␣
                    bounds_vert.y;\n" //
                + "}";

        final ShaderProgram shader = new ShaderProgram(vertexShader,
            fragmentShader);
        if (!shader.isCompiled())
            throw new IllegalArgumentException("Error␣compiling␣shader:␣
                " + shader.getLog());

        shader.bind();
        shader.setUniformf("alpha_threshold", 0.5f);

        return shader;
    }
}
```

Entity.java

```
package com.dar.freshmaze.entities;

import com.badlogic.gdx.graphics.g2d.Batch;
import com.badlogic.gdx.graphics.g2d.Sprite;
import com.badlogic.gdx.math.Matrix4;
import com.badlogic.gdx.math.Vector2;
import com.badlogic.gdx.physics.box2d.Body;
import com.badlogic.gdx.physics.box2d.World;
import com.dar.freshmaze.util.IsometricUtil;

public class Entity extends PhysActor {
    private Sprite sprite;
    private Vector2 spriteOffset;
    private SpriteKind spriteKind;
    private Matrix4 renderMatrix;
```

```
16
17
18      public Entity(World physWorld, Sprite sprite, Body body, Vector2
            spriteOffset, SpriteKind spriteKind, Vector2 spawnPos) {
19          super(physWorld, body);
20
21          this.sprite = sprite;
22          this.spriteOffset = spriteOffset;
23          this.spriteKind = spriteKind;
24          this.renderMatrix = spriteKind.getRenderMatrix();
25
26          setBounds(spawnPos.x, spawnPos.y, sprite.getWidth(), sprite.
                getHeight());
27          teleport(spawnPos);
28      }
29
30      public Sprite getSprite() {
31          return sprite;
32      }
33
34      @Override
35      protected void positionChanged() {
36          final Vector2 cartPos = new Vector2(getX() + spriteOffset.x,
                getY() + spriteOffset.y);
37          final Vector2 pos = spriteKind == SpriteKind.Isometric ?
                IsometricUtil.cartToIso(cartPos) : cartPos;
38          sprite.setPosition(pos.x, pos.y);
39
40          super.positionChanged();
41      }
42
43      @Override
44      public void draw(Batch batch, float alpha) {
45          setShaderSortHeight(batch, 0.0f);
46
47          batch.setTransformMatrix(renderMatrix);
48          sprite.draw(batch);
49
50          batch.flush();
51      }
52
53      protected void setShaderSortHeight(Batch batch, float offset) {
54          final Vector2 isoPos = IsometricUtil.cartToIso(new Vector2(getX
                () + getWidth() / 2, getY() + getHeight() / 2));
55          batch.getShader().setUniformf("height", isoPos.y + offset);
56      }
57
58      protected enum SpriteKind {
```

```
        Isometric,
        Transform;

        public Matrix4 getRenderMatrix() {
            switch (this) {
                case Isometric:
                    return new Matrix4().idt();
                case Transform:
                    return IsometricUtil.ISO_TRANSFORMATION_MATRIX;
                default:
                    throw new RuntimeException("Unhandled case in switch
                        ");
            }
        }
    }
}
```

PhysActor.java

```
package com.dar.freshmaze.entities;

import com.badlogic.gdx.math.Vector2;
import com.badlogic.gdx.physics.box2d.Body;
import com.badlogic.gdx.physics.box2d.World;
import com.badlogic.gdx.scenes.scene2d.Actor;

public class PhysActor extends Actor {
    private final World physWorld;

    private Body body;

    private boolean pendingDestroy = false;
    private boolean isDestroyed = false;

    public PhysActor(World physWorld, Body body) {
        this.physWorld = physWorld;
        this.body = body;

        body.setUserData(this);
    }

    public Body getBody() {
        return body;
    }

    public World getPhysWorld() {
         return physWorld;
    }
```

```
30
31      public boolean isDestroyed() {
32           return isDestroyed;
33      }
34
35      public void teleport(Vector2 pos) {
36          body.setTransform(pos, 0.0f);
37          updatePosition();
38      }
39
40      public void destroy() {
41          pendingDestroy = true;
42      }
43
44      @Override
45      public boolean remove() {
46          destroy();
47
48          return super.remove();
49      }
50
51      @Override
52      public void act(float delta) {
53          super.act(delta);
54
55          if (pendingDestroy) {
56              if (body != null)
57                  physWorld.destroyBody(body);
58              remove();
59
60              pendingDestroy = false;
61              isDestroyed = true;
62
63              return;
64          }
65
66          updatePosition();
67      }
68
69      private void updatePosition() {
70          setPosition(getBody().getPosition().x, getBody().getPosition().y
                );
71      }
72  }
```

RectIndicator.java

```
1  package com.dar.freshmaze.indicator;
```

```
2
3  import com.badlogic.gdx.graphics.Color;
4  import com.badlogic.gdx.graphics.Texture;
5  import com.badlogic.gdx.graphics.g2d.Batch;
6  import com.badlogic.gdx.scenes.scene2d.Actor;
7
8  public class RectIndicator extends Actor {
9      private final Texture texture;
10
11     private Color backgroundColor = Color.BLACK;
12     private Color indicatorColor = Color.WHITE;
13
14     private FloatRangeBinder valueBinder;
15
16     public RectIndicator(FloatRangeBinder valueBinder) {
17         super();
18
19         texture = new Texture("rectangle.png");
20
21         this.valueBinder = valueBinder;
22     }
23
24     public void setBackgroundColor(Color newBackgroundColor) {
25         backgroundColor = newBackgroundColor;
26     }
27
28     public void setIndicatorColor(Color newIndicatorColor) {
29         indicatorColor = newIndicatorColor;
30     }
31
32     @Override
33     public void draw(Batch batch, float parentAlpha) {
34         batch.setColor(backgroundColor);
35         batch.draw(texture, getX(), getY(), getWidth(), getHeight());
36         batch.setColor(indicatorColor);
37         batch.draw(texture, getX(), getY(), getWidth() * getNormValue(),
               getHeight());
38     }
39
40     protected float getNormValue() {
41         return valueBinder.getValue() / valueBinder.getMaxValue();
42     }
43
44     public interface FloatRangeBinder {
45         float getValue();
46         float getMaxValue();
47     }
48 }
```

WorldContactListener.java

```java
package com.dar.freshmaze.world;

import com.badlogic.gdx.physics.box2d.*;
import com.dar.freshmaze.entities.Bob;

public class WorldContactListener implements ContactListener {
    @Override
    public void beginContact(Contact contact) {
        Fixture fixA = contact.getFixtureA();
        Fixture fixB = contact.getFixtureB();
        if(fixA.isSensor() == fixB.isSensor())
            return;

        final Object first = fixA.getBody().getUserData();
        final Object second = fixB.getBody().getUserData();
        if (first != null && second != null) {
            if (first instanceof Bob)
                ((Bob)first).addObjectInRadius(second);
            else if (second instanceof Bob)
                ((Bob)second).addObjectInRadius(first);
        }
    }

    @Override
    public void endContact(Contact contact) {
        Fixture fixA = contact.getFixtureA();
        Fixture fixB = contact.getFixtureB();
        if(fixA.isSensor() == fixB.isSensor())
            return;

        final Object first = fixA.getBody().getUserData();
        final Object second = fixB.getBody().getUserData();
        if (first != null && second != null) {
            if (first instanceof Bob)
                ((Bob)first).removeObjectInRadius(second);
            else if (second instanceof Bob)
                ((Bob)second).removeObjectInRadius(first);
        }
    }

    @Override
    public void preSolve(Contact contact, Manifold oldManifold) {

    }

    @Override
    public void postSolve(Contact contact, ContactImpulse impulse) {
```

```
48
49      }
50  }
```

DepthSortedStage.java

```
1   package com.dar.freshmaze.graphics;
2
3   import com.badlogic.gdx.Gdx;
4   import com.badlogic.gdx.graphics.Camera;
5   import com.badlogic.gdx.graphics.GL20;
6   import com.badlogic.gdx.graphics.g2d.Batch;
7   import com.badlogic.gdx.graphics.glutils.ShaderProgram;
8   import com.badlogic.gdx.math.Vector2;
9   import com.badlogic.gdx.scenes.scene2d.Stage;
10  import com.badlogic.gdx.utils.viewport.Viewport;
11
12  /**
13   * Modified stage, that renders object using OpenGL depth test
14   */
15  public class DepthSortedStage extends Stage {
16      private Vector2 verticalViewBounds;
17
18      public DepthSortedStage() {
19          super();
20          init();
21      }
22      /** Creates a stage with the specified viewport. The stage will use
            its own {@link Batch} which will be disposed when the stage
23       * is disposed. */
24      public DepthSortedStage(Viewport viewport) {
25          super(viewport);
26          init();
27      }
28
29      /** Creates a stage with the specified viewport and batch. This can
            be used to specify an existing batch or to customize which
30       * batch implementation is used.
31       * @param batch Will not be disposed if {@link #dispose()} is called
             , handle disposal yourself. */
32      public DepthSortedStage(Viewport viewport, Batch batch) {
33          super(viewport, batch);
34          init();
35      }
36
37      private void init() {
38          getBatch().setShader(createShader());
39      }
```

```
40
41      public Vector2 getVerticalViewBounds() {
42          return verticalViewBounds;
43      }
44
45      public void setVerticalViewBounds(Vector2 newVerticalViewBounds) {
46          verticalViewBounds = newVerticalViewBounds;
47      }
48
49      public void shaderSetVerticalViewBounds() {
50          final Vector2 bounds = getVerticalViewBounds();
51          getBatch().getShader().bind();
52          getBatch().getShader().setUniform2fv("bounds_vert", new float[]
                { bounds.x, bounds.y }, 0, 2);
53      }
54
55      @Override
56      public void draw () {
57          Camera camera = getViewport().getCamera();
58          camera.update();
59
60          if (!getRoot().isVisible()) return;
61
62          Batch batch = getBatch();
63          batch.setProjectionMatrix(camera.combined);
64          batch.begin();
65
66          Gdx.gl.glEnable(GL20.GL_DEPTH_TEST);
67          Gdx.gl.glDepthMask(true);
68          Gdx.gl.glDepthFunc(GL20.GL_LESS);
69
70          getRoot().draw(batch, 1);
71          batch.end();
72
73          // if (debug) drawDebug();
74      }
75
76      private static ShaderProgram createShader() {
77          final String vertexShader = "attribute␣vec4␣" + ShaderProgram.
                POSITION_ATTRIBUTE + ";\n" //
78                  + "attribute␣vec4␣" + ShaderProgram.COLOR_ATTRIBUTE + "
                        ;\n" //
79                  + "attribute␣vec2␣" + ShaderProgram.TEXCOORD_ATTRIBUTE +
                        "0;\n" //
80                  + "uniform␣mat4␣u_projTrans;\n" //
81                  + "varying␣vec4␣v_color;\n" //
82                  + "varying␣vec2␣v_texCoords;\n" //
83                  + "\n" //
```

```
84                  + "void␣main()\n" //
85                  + "{\n" //
86                  + "␣␣␣␣v_color␣=␣" + ShaderProgram.COLOR_ATTRIBUTE + ";\n
                        " //
87                  + "␣␣␣␣v_color.a␣=␣v_color.a␣*␣(255.0/254.0);\n" //
88                  + "␣␣␣␣v_texCoords␣=␣" + ShaderProgram.TEXCOORD_ATTRIBUTE
                        + "0;\n" //
89                  + "␣␣␣␣gl_Position␣=␣␣u_projTrans␣*␣" + ShaderProgram.
                        POSITION_ATTRIBUTE + ";\n" //
90                  + "}\n";
91          final String fragmentShader = "#ifdef␣GL_ES\n" //
92                  + "#define␣LOWP␣lowp\n" //
93                  + "precision␣mediump␣float;\n" //
94                  + "#else\n" //
95                  + "#define␣LOWP␣\n" //
96                  + "#endif\n" //
97                  + "varying␣LOWP␣vec4␣v_color;\n" //
98                  + "varying␣vec2␣v_texCoords;\n" //
99                  + "uniform␣sampler2D␣u_texture;\n" //
100                 + "uniform␣float␣height;\n" //
101                 + "uniform␣vec2␣bounds_vert;\n" //
102                 + "uniform␣float␣alpha_threshold;\n"
103                 + "void␣main()\n" //
104                 + "{\n" //
105                 + "␣␣vec4␣tex␣=␣texture2D(u_texture,␣v_texCoords);\n" //
106                 + "␣␣if(tex.a␣<␣alpha_threshold)\n" //
107                 + "␣␣␣␣␣␣␣discard;\n"//
108                 + "\n" //
109                 + "␣␣gl_FragColor␣=␣v_color␣*␣tex;\n"
110                 + "␣␣gl_FragDepth␣=␣(height␣-␣bounds_vert.x)␣/␣
                        bounds_vert.y;\n" //
111                 + "}";
112
113         final ShaderProgram shader = new ShaderProgram(vertexShader,
                fragmentShader);
114         if (!shader.isCompiled())
115             throw new IllegalArgumentException("Error␣compiling␣shader:␣
                    " + shader.getLog());
116
117         shader.bind();
118         shader.setUniformf("alpha_threshold", 0.5f);
119
120         return shader;
121     }
122 }
```

ScreenTransition.java

```java
package com.dar.freshmaze.ui;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.Batch;
import com.badlogic.gdx.graphics.glutils.ShaderProgram;
import com.badlogic.gdx.scenes.scene2d.Actor;

public class ScreenTransition extends Actor {
    private static final Texture texture = new Texture(Gdx.files.
        internal("rectangle.png"));
    private static final ShaderProgram shader = createShader();

    private Color color = Color.BLACK;
    private final float speed;
    private final float duration;
    private float time = 0.0f;

    private boolean fadeOut;
    private boolean isFrozen = false;

    private final TransitionCallback callback;

    public ScreenTransition(float speed, float duration, boolean fadeOut
        ) {
        this(speed, duration, fadeOut, null);
    }

    public ScreenTransition(float speed, float duration, boolean fadeOut
        , TransitionCallback callback) {
        this.speed = speed;
        this.duration = duration;
        this.fadeOut = fadeOut;
        this.callback = callback;

        shader.bind();
        shader.setUniformf("fade_mult", fadeOut ? -1.0f : 1.0f);
    }

    public void setColor(Color newColor) {
        color = newColor;
    }

    public boolean isFrozen() {
        return isFrozen;
    }

```

```
46      public void setIsFrozen(boolean newIsFrozen) {
47          isFrozen = newIsFrozen;
48      }
49
50      @Override
51      public void act(float delta) {
52          super.act(delta);
53
54          if (isFrozen)
55              return;
56
57          if (time >= duration) {
58              if (fadeOut)
59                  remove();
60
61              if (callback != null)
62                  callback.onComplete();
63
64              return;
65          }
66
67          time += delta * speed;
68      }
69
70      @Override
71      public void draw(Batch batch, float parentAlpha) {
72          final ShaderProgram oldShader = batch.getShader();
73          batch.setShader(shader);
74
75          batch.getShader().setUniformf("time", time);
76          batch.setColor(color);
77          batch.draw(texture, 0.0f, 0.0f, getStage().getWidth(), getStage
                ().getHeight());
78          batch.flush();
79
80          batch.setShader(oldShader);
81      }
82
83      private static ShaderProgram createShader() {
84          final String vertexShader = "attribute␣vec4␣" + ShaderProgram.
                POSITION_ATTRIBUTE + ";\n" //
85                  + "attribute␣vec4␣" + ShaderProgram.COLOR_ATTRIBUTE + "
                        ;\n" //
86                  + "attribute␣vec2␣" + ShaderProgram.TEXCOORD_ATTRIBUTE +
                        "0;\n" //
87                  + "uniform␣mat4␣u_projTrans;\n" //
88                  + "varying␣vec4␣v_color;\n" //
89                  + "varying␣vec2␣v_texCoords;\n" //
```

```
90                  + "\n" //
91                  + "void␣main()\n" //
92                  + "{\n" //
93                  + "␣␣␣v_color␣=␣" + ShaderProgram.COLOR_ATTRIBUTE + ";\n
                        " //
94                  + "␣␣␣v_color.a␣=␣v_color.a␣*␣(255.0/254.0);\n" //
95                  + "␣␣␣v_texCoords␣=␣" + ShaderProgram.TEXCOORD_ATTRIBUTE
                        + "0;\n" //
96                  + "␣␣␣gl_Position␣=␣␣u_projTrans␣*␣" + ShaderProgram.
                        POSITION_ATTRIBUTE + ";\n" //
97                  + "}\n";
98          final String fragmentShader = "#ifdef␣GL_ES\n" //
99                  + "#define␣LOWP␣lowp\n" //
100                 + "precision␣mediump␣float;\n" //
101                 + "#else\n" //
102                 + "#define␣LOWP␣\n" //
103                 + "#endif\n" //
104                 + "varying␣LOWP␣vec4␣v_color;\n" //
105                 + "varying␣vec2␣v_texCoords;\n" //
106                 + "uniform␣sampler2D␣u_texture;\n" //
107                 + "uniform␣float␣time;\n" //
108                 + "uniform␣float␣duration;\n" //
109                 + "uniform␣float␣fade_mult;\n" //
110                 + "void␣main()\n" //
111                 + "{\n" //
112                 + "␣␣vec2␣uv␣=␣v_texCoords;\n" //
113                 + "␣␣uv␣=␣(uv␣-␣vec2(0.5))␣*␣2.0;\n" //
114                 + "␣␣\n" //
115                 + "␣␣vec4␣tex␣=␣texture2D(u_texture,␣uv);\n" //
116                 + "␣␣tex.a␣=␣step(fade_mult␣*␣cos(time),␣uv.x␣*␣uv.y);\n
                        " //
117                 + "\n" //
118                 + "␣␣gl_FragColor␣=␣v_color␣*␣tex;\n"
119                 + "}";
120
121         final ShaderProgram shader = new ShaderProgram(vertexShader,
                fragmentShader);
122         if (!shader.isCompiled())
123             throw new IllegalArgumentException("Error␣compiling␣shader:␣
                    " + shader.getLog());
124
125         return shader;
126     }
127
128     public interface TransitionCallback {
129         void onComplete();
130     }
131 }
```

Перелік ілюстрацій