UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

UNIVERSITI TEKNOLOGI MALAYSIA

SEM 2 2023/2024

SECJ1023 - PROGRAMMING TECHNIQUES II

SECTION 03

PROJECT DELIVERABLE 2

TITLE : PROBLEM ANALYSIS AND DESIGN

PREPARED BY : GROUP PIXCREW

NAME	MATRIC NO
LUBNA AL HAANI BINTI RADZUAN	A23CS0107
NUR FIRZANA BINTI BADRUS HISHAM	A23CS0156
NURUL ASYIKIN BINTI KHAIRUL ANUAR	A23CS0162
ANIS SAFIYYA BINTI JANAI	A23CS0049

PREPARED FOR:

DR. JUMAIL BIN TALIBA

Link brainstorming session => Discussion Video.MOV

Problem Analysis

Classes and Objects:

1. **Class :** Point
   Attributes : int x, int y
   Method : Point(), Point(int _x, int _y), void setx(int _x), void sety(int _y), int getx(), int gety()

2. **Class :** Background
   Attributes : Point locatBg, int height, int length, int size, int color
   Method : Background(), Background(int h, int l, int color), Background(int _x, int _y, int size, int color), draw()

3. **Class :** Food
   Attributes : Point locatFood, int size, int color, int dx, int dy
   Method : Food(), Food(int _x, int _y, int size, int color), void setHorizontalLocat(int _dy), void xetVerticalMove(int _dx), int getYLocat(), void setVerticalMove(int _dx), int getYLocat(), void MoveFaster(), void draw(), void undraw(), void FoodHit(HitArea h)

4. **Class :** HitArea
   Attributes : int yAxis[], int xAxis[]
   Method : HitArea(), int gotHit(), void setY(), void setsetX(), int getY()

5. **Class :** Floor
   Attributes : Point locatFloor, int height, int length, int color
   Method : Floor(), Floor(int _x, int _y, int h, int l, int color), void setColor(), int getx(), int gotHit(), draw()

6. **Class :** Player
   Attributes : Point locatPlayer, Score playerScore, Food foodScore, int dx
   Method : Player(), Player(int _x, int _y), void setLocatPlayer(int _x, int _y), void setHorizontalMove(int _dx), reverseHorizontal(int _dx), int getx(), int gotHit(), void draw(), void undraw()

7. **Class :** Score
   Attributes : int score
   Method : void getScore(), int getScore(), void printScore()

Relationship

1. **Inheritance**
   Class: Point, Background, Food, Player and Floor

   The Background, Food, Player and Floor classes inherit from the Point class. The fundamental coordinate system's x and y locations are provided by the Point class. There are methods in this class to set and retrieve these coordinate values. The inheritance connections allow the classes Background, Food, Floor, and Player to make use of the Point class's coordinate capabilities while extending it with attributes and methods relevant to their game roles.

2. **Polymorphism**
   Class: HitArea, Floor and Player

   Both the Player and Floor classes derive from the HitArea class, which allows for polymorphism behaviour. This means that both Player and Floor implement HitArea's virtual methods, including gotHit(), setY(), setX(), and getY(). This polymorphism connection allows objects of the Player and Floor types to be handled as HitArea objects, allowing for their flexible and interchangeable use in situations requiring hit detection. For example, a method that checks for hits can be applied on an array of HitArea objects, independent of whether they are Player or Floor instances, boosting code reuse and flexibility.

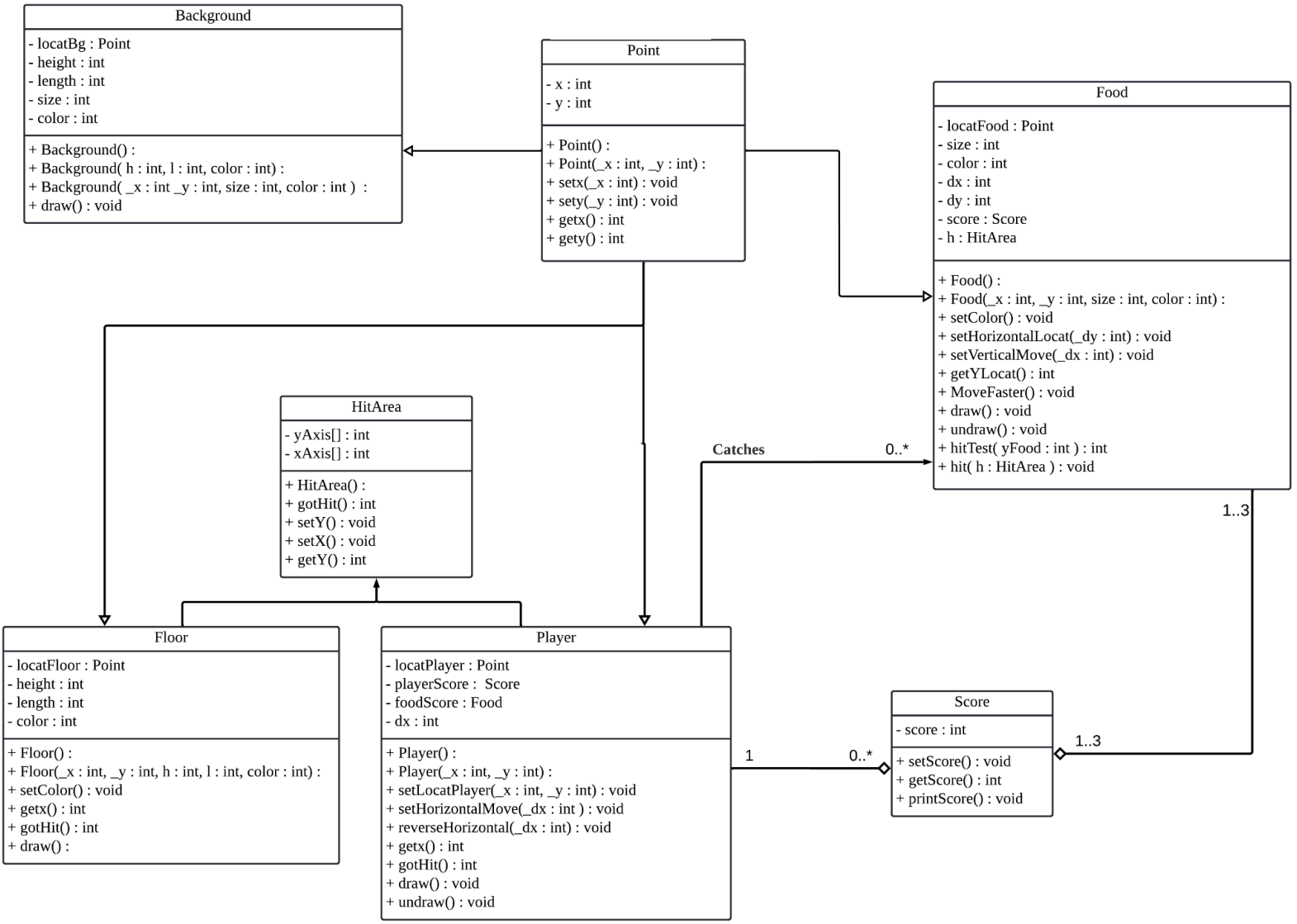
3. **Associations**
   Class: Player and Food

   In this game, the Player and Food classes form an association. These classes are related because the player, who is shown as a cat in the game, catches the food. This one-to-many relationship means that the player can catch numerous food items that are scattered throughout the game environment.

4. **Aggregation**
   Class: Score, Food, and Player

   Aggregation relationships are applied by the Score class with the Food and Player classes. There will be more than one Food item linked to a Score, although Food items can exist separately from Scores. The aggregation relationship indicates that different foods may have an effect on or relate to the game's score. For instance, the game consists of three types of foods, and each type would count differently toward the final score. Meanwhile, class Score is related to Player objects, but the Player objects can exist independently of the Score.

UML Diagram


Background
- locatBg : Point
- height : int
- length : int
- size : int
- color : int
+ Background() :
+ Background(h : int, l : int, color : int) :
+ Background(_x : int _y : int, size : int, color : int) :
+ draw() : void
Point
- x : int
- y : int
+ Point() :
+ Point(_x : int, _y : int) :
+ setx(_x : int) : void
+ sety(_y : int) : void
+ getx() : int
+ gety() : int
Food
- locatFood : Point
- size : int
- color : int
- dx : int
- dy : int
- score : Score
- h : HitArea
+ Food() :
+ Food(_x : int, _y : int, size : int, color : int) :
+ setColor() : void
+ setHorizontalLocat(_dy : int) : void
+ setVerticalMove(_dx : int) : void
+ getYLocat() : int
+ MoveFaster() : void
+ draw() : void
+ undraw() : void
+ hitTest(yFood : int) : int
+ hit(h : HitArea) : void
HitArea
- yAxis[] : int
- xAxis[] : int
+ HitArea() :
+ gotHit() : int
+ setY() : void
+ setX() : void
+ getY() : int
Catches
0..*
1..3
Floor
- locatFloor : Point
- height : int
- length : int
- color : int
+ Floor() :
+ Floor(_x : int, _y : int, h : int, l : int, color : int) :
+ setColor() : void
+ getx() : int
+ gotHit() : int
+ draw() :
Player
- locatPlayer : Point
- playerScore : Score
- foodScore : Food
- dx : int
+ Player() :
+ Player(_x : int, _y : int) :
+ setLocatPlayer(_x : int, _y : int) : void
+ setHorizontalMove(_dx : int) : void
+ reverseHorizontal(_dx : int) : void
+ getx() : int
+ gotHit() : int
+ draw() : void
+ undraw() : void
1
0..*
Score
- score : int
+ setScore() : void
+ getScore() : int
+ printScore() : void
1..3