Name __Andrew Plum__ Mark ______________________/50

1. Brief introduction __/3

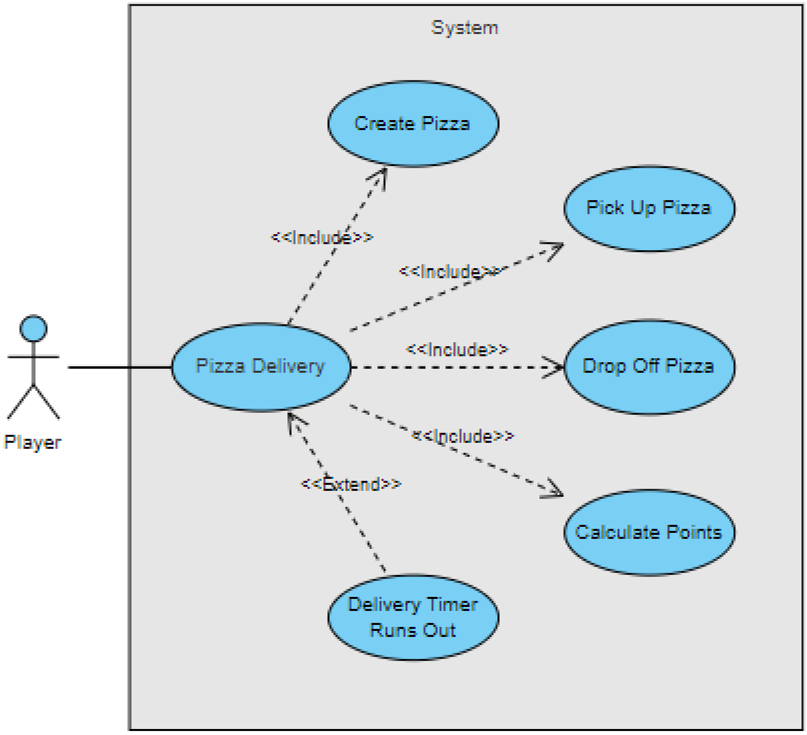
Midnight Slice Madness is a pizza delivery game with a horror spin to it. In the game, the feature that I am developing is part of the main gameplay loop. The main gameplay loop consists of pizza delivery where the main character will create pizza objects that can be picked up and dropped by the character so they can be delivered to their destination for points. There will be different pizza types for the pizza objects which will involve a decoration design pattern. I will for sure be developing the pizza request, creation, and pick up features. I will not start working on the pizza drop off and calculate points feature until later because these features will be there for Zoe to do if she needs to complete any code requirements she missed in developing her feature like implementing a pattern.

2. Use case diagram with scenario __14

Use Case Diagrams

Scenarios

First Scenario (Use Case Diagram):



Name: Pizza delivery
Summary: The player creates and picks up a pizza so it can be delivered.
Actors: The player
Preconditions: Player has installed and started game.
Basic sequence:

Step 1: Player presses button to create pizza object.

Step 2: Player picks up pizza object.

Step 3: Player travels to delivery destination with pizza object.

Step 4: Player drops pizza object off at delivery destination.

Exceptions:

Step 3: Delivery timer runs out: the player can no longer deliver the pizza to the specified delivery destination

Post conditions: The pizza is delivered, and the player receives points.

Priority: 1*

ID: A01

*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

3. Data Flow diagram(s) from Level 0 to process description for your feature ______14

Data Flow Diagrams

Context Diagram:

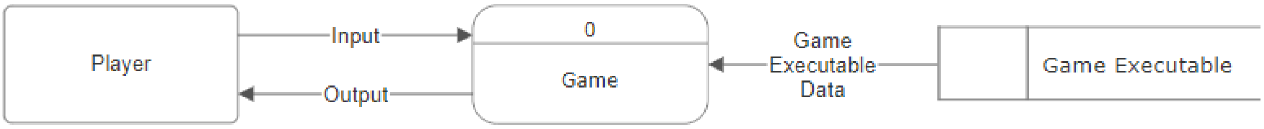


Diagram 0:

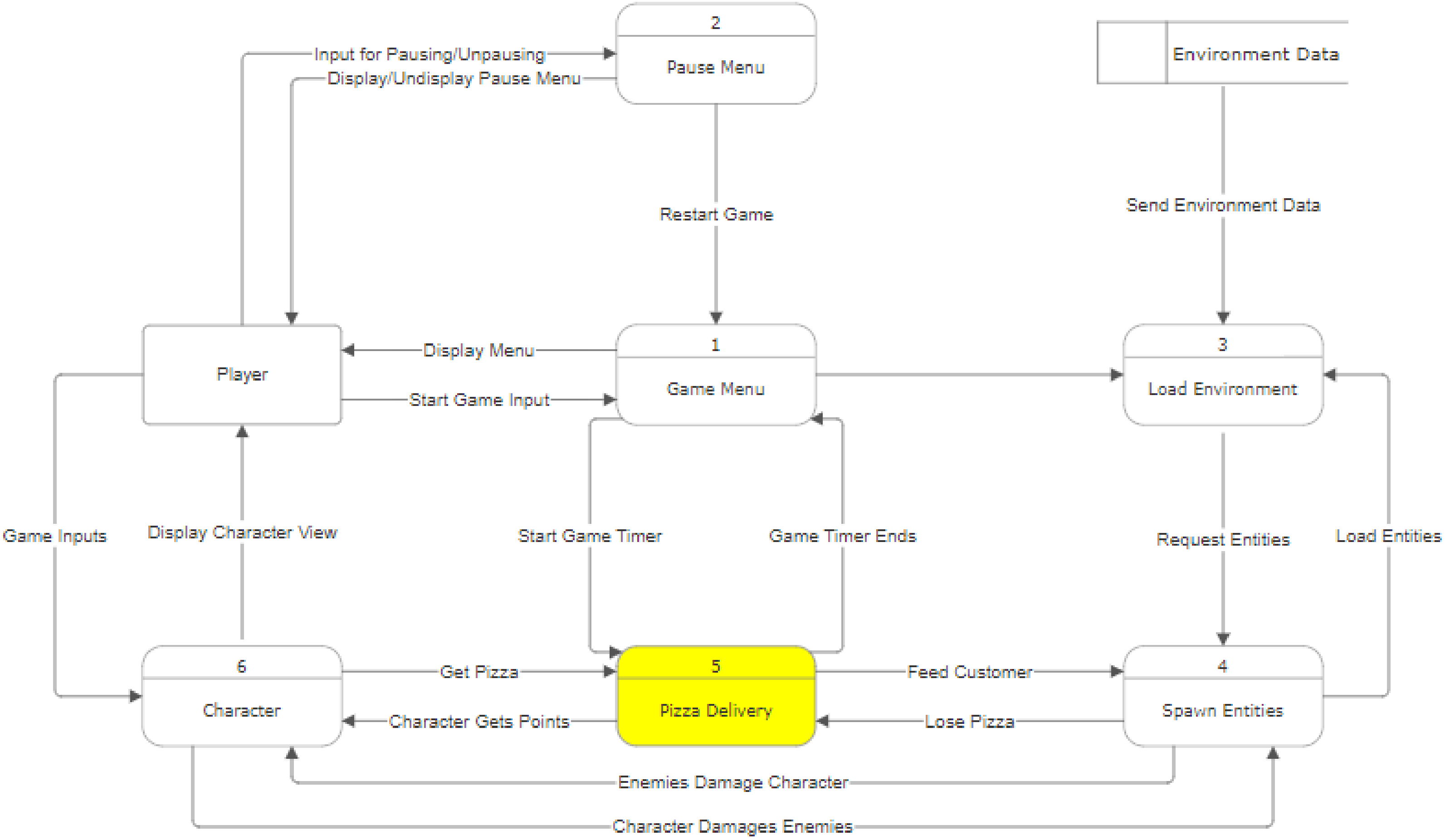
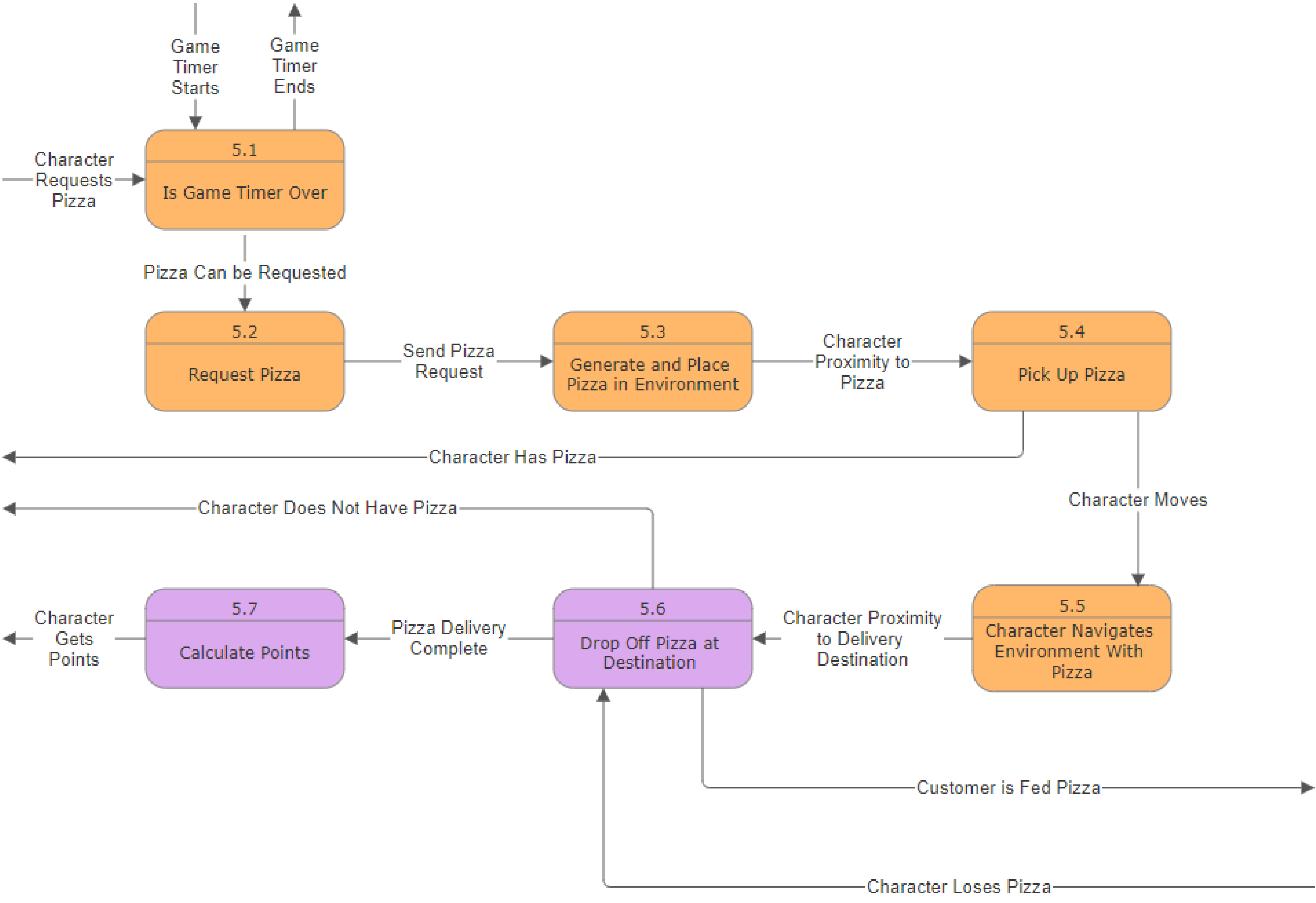

2
Pause Menu
Environment Data
Input for Pausing/Unpausing
Display/Undisplay Pause Menu
Restart Game
Send Environment Data
Player
Display Menu
Start Game Input
1
Game Menu
3
Load Environment
Game Inputs
Display Character View
Start Game Timer
Game Timer Ends
Request Entities
Load Entities
6
Character
Get Pizza
Character Gets Points
5
Pizza Delivery
Feed Customer
Lose Pizza
4
Spawn Entities
Enemies Damage Character
Character Damages Enemies

Diagram 5:



Key:

Orange Processes: These are the processes I will be responsible for.

Purple Processes: These are the processes I will possibly be responsible for if Zoe does not need to do them to meet her individual coding requirements.

Process Descriptions

Process Description Overview: The pizza delivery process involves various steps to ensure that the character successfully requests, receives, and delivers a pizza to the customer. The process begins with the game timer running and determining if the game timer is over, potentially allowing the player to request a pizza if the game is ongoing (5.1). If the player requests a pizza (5.2), a pizza is generated by the system and placed in the environment (5.3). The system then calculates the character's proximity to the pizza to determine if the character can pick up the pizza; the player picks up the pizza if close enough in proximity (5.4). The character then navigates the environment while holding the pizza (5.5). Once the pizza is successfully delivered to the destination, the pizza delivery process is complete (5.6), and the system calculates points to reward the player for successfully completing the delivery (5.7).

Process 5.1 Pseudocode:

```
gameTimerIsNotOver()
{
      if(gameTimer > 0)
      {
         return true;
      }
      else
      {
         return false;
      }
}
```

Process 5.2 Pseudocode:

```
requestPizza()
{
    if(gameTimerIsNotOver())
    {
        generateAndPlacePizzaInEnvironment();
    }
}
```

Process 5.3 Pseudocode:

```
generateAndPlacePizzaInEnvironment()
{
    pizza = createPizzaObject();
    placePizzaInEnvironment(pizza);
}
```

Process 5.4 Pseudocode:

```
pickUpPizza()
{
    if(characterInProximityOfPizza())
    {
        character.pickUp(pizza);
    }
}
```

Process 5.5 Pseudocode:

```
characterNavigatesEnvironmentWithPizza()
{
    character.moveTowards(deliveryDestination);
}
```

Process 5.6 Pseudocode:

```
characterInProximityOfDeliveryDestination()
{
    distance =
    calculateDistance(characterPosition,deliveryDestination);
    return distance <= deliveryDestinationProximityThreshold;
}

dropOffPizzaAtDestination()
{
    if(characterInProximityOfDeliveryDestination())
    {
        character.dropOff(pizza);
        calculateAndRewardCharacterPoints();
    }
}
```

Process 5.7 Pseudocode:

```
calculateAndRewardCharacterPoints()
{
    points = calculatePointsBasedOnDeliverySpeed();
    updatePlayerScore(points);
}
```

4. Acceptance Tests ______9

Overview: The pizza delivery feature involves processes from requesting a pizza to delivering it successfully. Various inputs and outputs are tested to ensure the seamless functioning of the gameplay.

1. **Request Pizza**
   - **Input:** Game timer not expired, and player interacts with pizza request button.
   - **Expected Output:** No pizza is generated and placed in the game environment.
   - **Boundary Cases:** Test with different game timer durations.
2. **Request Pizza (Opposite)**
   - **Input:** Game timer expired, and player interacts with pizza request button.
   - **Expected Output:** A new pizza is not generated and placed in the game environment.
   - **Boundary Cases:** Test with various game timer expiration conditions.
3. **Pick Up Pizza**
   - **Input:** Character is in proximity to the pizza.
   - **Expected Output:** Character can pick up the pizza.
   - **Boundary Cases:** Test with different proximity thresholds.
4. **Pick Up Pizza (Opposite)**
   - **Input:** Character is not in proximity to a pizza.
   - **Expected Output:** Character cannot pick up a pizza.
   - **Boundary Cases:** Test with different proximity thresholds.
5. **Navigate Environment with Pizza**
   - **Input:** Character moves while holding the pizza.
   - **Expected Output:** Character moves towards the delivery destination with the pizza.
   - **Boundary Cases:** Test by moving character with pizza all over the map.
6. **Drop Off Pizza**
   - **Input:** Character is with pizza and in proximity to the delivery destination.

- **Expected Output:** Pizza is dropped off, and points are calculated.
- **Boundary Cases:** Test with different distances from delivery destination conditions.

7. **Drop Off Pizza (Opposite)**

- **Input:** Character is with pizza and not in proximity to the delivery destination.
- **Expected Output:** Pizza is not dropped off, and points are not calculated.
- **Boundary Cases:** Test with different distances from delivery destinations.

8. **Calculate and Reward Character Points**

- **Input:** Specific delivery time and successful drop-offs.
- **Expected Output:** Points are calculated accurately based on the delivery time.
- **Boundary Cases:** Test with variation in delivery times.

5. Timeline ________/10

Work items

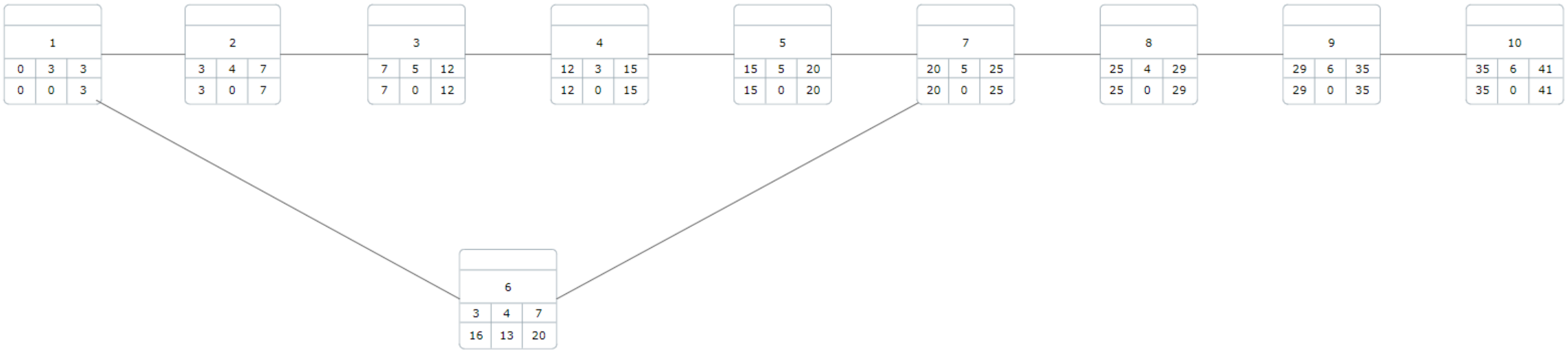
Estimated Total Hours: 45 hours

This schedule outlines the individual tasks, their estimated durations, and their dependencies, allowing for efficient development of the pizza delivery feature.

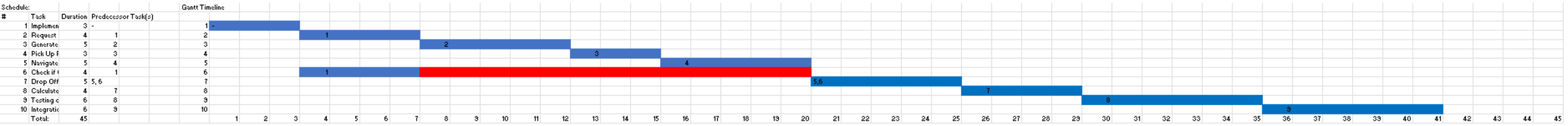
Task	Duration (Hours)	Predecessor Task(s)
1. Implement Game Timer	3	-
2. Request Pizza Function	4	1
3. Generate and Place Pizza in Environment	5	2
4. Pick Up Pizza	3	3
5. Navigate Environment with Pizza	5	4
6. Check if Character is in Proximity of Delivery Environment	4	1

7. Drop Off Pizza at Destination	5	5, 6
8. Calculate and Reward Character Points	4	7
9. Testing of Individual Responsibilities	6	8
10. Integration of Individual Responsibilities with Team's Work	6	9
TOTAL	45	-

Pert Diagram



Gantt Timeline


Schedule:

#	Task	Duration	Predecessor Task(s)
1	Implemen	3	-
2	Request	4	1
3	Generate	5	2
4	Pick Up F	3	3
5	Navigate	5	4
6	Check if (	4	1
7	Drop Off	5	5, 6
8	Calculate	4	7
9	Testing c	6	8
10	Integratic	6	9
	Total:	45	

***Note: Zoom in for clearer view**