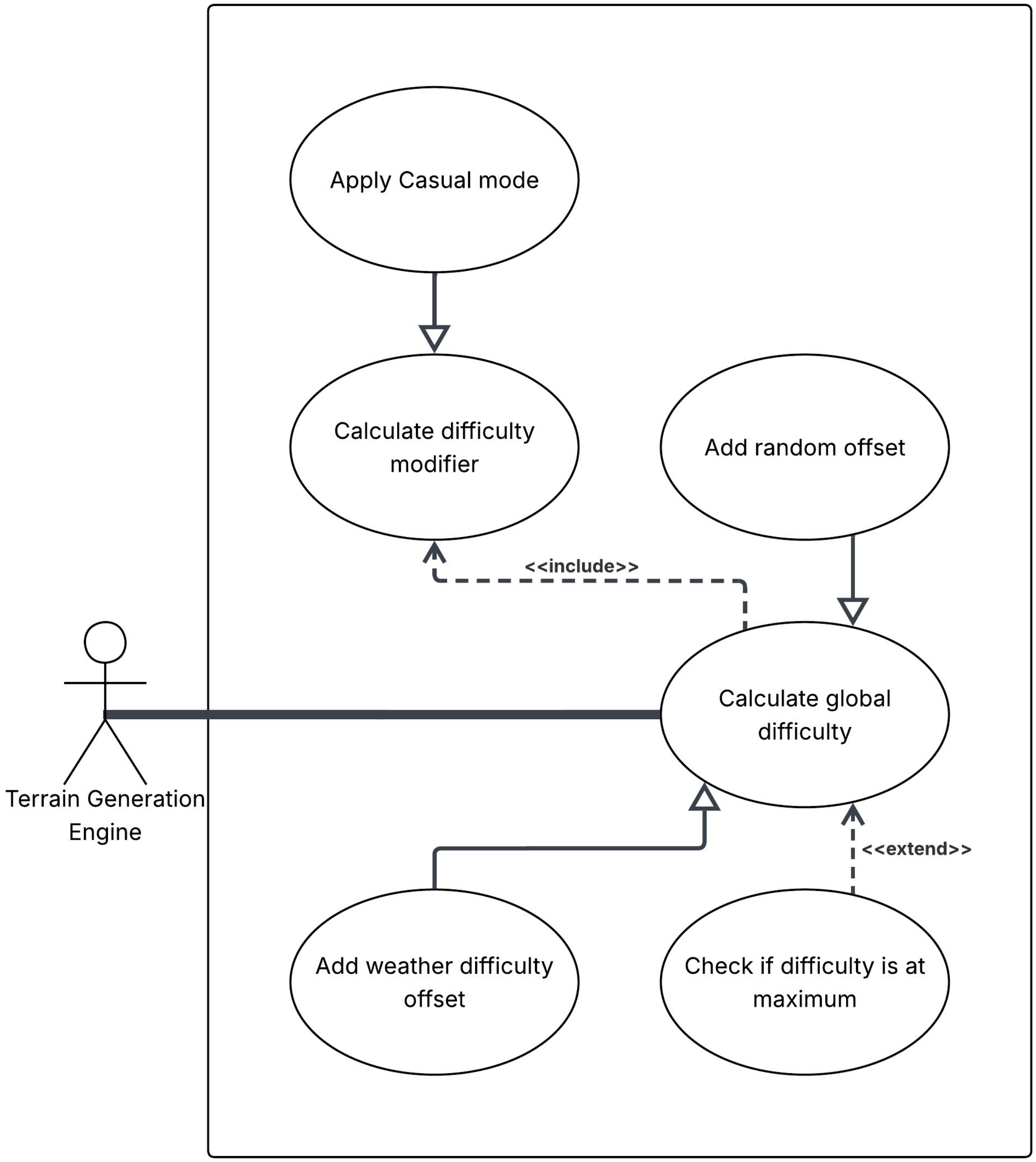
Name: **Johnathan Van Vliet** Mark _______________________/50

1. Brief introduction __/3

This Use Case Diagram introduces the global difficulty level and is used by the terrain generation engine to generate rock positions. As the boat drifts during fishing, these rock positions will be used to determine if the boat is damaged by rocks and the extent to which it is damaged.

2. Use case diagram with scenario __14

Use Case Diagrams


Apply Casual mode
Calculate difficulty modifier
Add random offset
<<include>>
Calculate global difficulty
Terrain Generation Engine
<<extend>>
Add weather difficulty offset
Check if difficulty is at maximum

Scenarios

Name: Calculate Global Difficulty

Summary: Global difficulty is calculated using several factors

Actors: Terrain Generation Engine

Preconditions: Boat must be active. This occurs upon boat activation

Basic sequence:

Step 1: Start with a base global difficulty of 1.0

Step 2: Add the offset from the current weather

Step 3: Add a randomly generated offset

Step 4: Multiply the difficulty by the current player difficulty offset

Exceptions:

Step 4: If difficulty is greater than the maximum, set difficulty to the maximum value.

Post conditions:

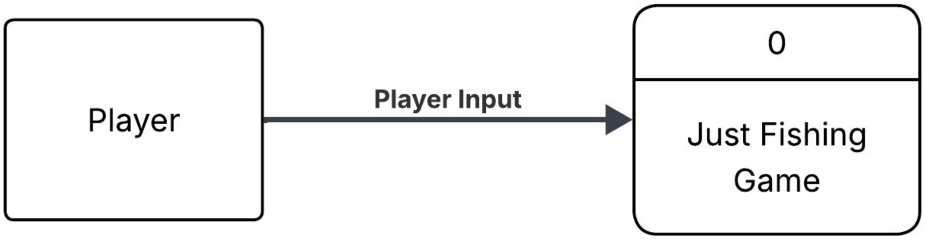
Priority: 1*

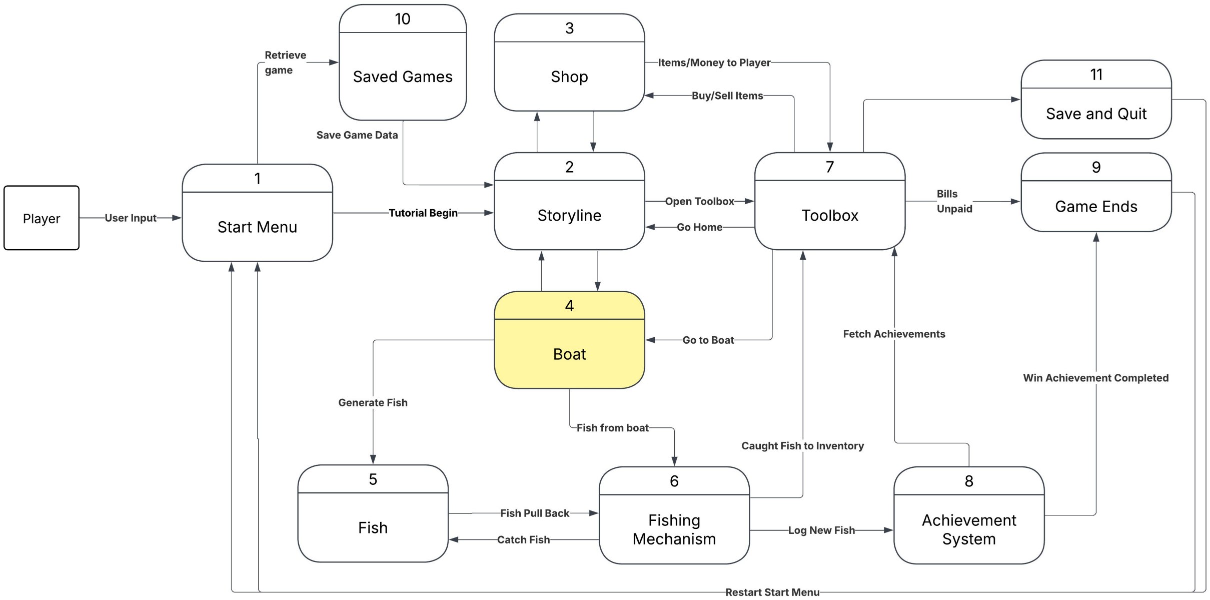
ID: DIFF_SET

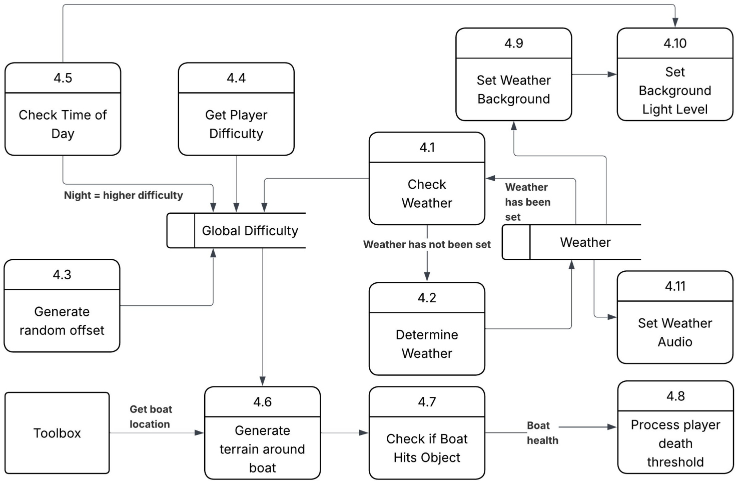
*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

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

Data Flow Diagrams







Process Descriptions

```
include RandomNumberGenerator as RNG;

string DetermineWeather()
{
        LIST string possible_weather =
                {"clear", "rainy", "stormy", "cloudy"};

        float weatherValue = RNG.GenerateNumber(float, 0.00, 1.00);
        if (weatherValue < 0.10)
                return "stormy"
        else if (weatherValue < 0.30)
        return "rainy"
    else if (weatherValue < 0.60)
        return "cloudy"
    else
        return "clear"
}

void SetWeatherSigns(current_weather, light_level)
{
    Scene.SetBackground(current_weather, light_level);
    Scene.PlayAudio(current_weather);
}

float GenerateGlobalDifficulty(current_weather, time_of_day, player_difficulty)
{
    float global_difficulty = 1.0;
    float time_difficulty = 1.0;

    DICT weather_difficulties = {
        "stormy": 2.0
        "rainy": 1.5
        "cloudy": 1.0
        "clear": 0.75
    }

    DICT player_difficulties = {
        "casual": 1.0
        "hard": 2.0
        "drbc": 0.0
    };

    # In 24 hour time
    if (time_of_day >= 20 or time_of_day < 6)
        time_difficulty = 2.0;

    global_difficulty =
        weather_difficulties[current_weather] +
        player_difficulties[player_difficulty] +
        time_difficulty +
        RNG.GenerateNumber(float, 0.00, 0.25);

    return global_difficulty
```

```
}

TerrainObject GenerateTerrain(boat_position, global_difficulty)
{
        int min_rocks = 0;
        int max_rocks = global_difficulty * 10;
        int num_rocks = min_rocks;

        DICT locations = {
                north_sea: 2
                south_sea: 3
                shallow_sea: 1
                deep_sea: 4
        }

        int highest_num_rocks = num_rocks;
        for (locations[boat_position])
        {
                do:
                        num_rocks = RNG.GenerateNumber(int, min_rocks, max_rocks);
                until:
                        num_rocks > highest_num_rocks;

                highest_num_rocks = num_rocks;
        }

        ARRAY TerrainObject rocks[num_rocks];
        for (rock in rocks)
        {
                int new_rock_location

                do:
                        new_rock_location = RNG.GenerateNumber(signed float, -20, 20);
                until:
                        new_rock_location not in rock_locations;

                rock_locations[rock].rock_location = new_rock_location;
        }

        return rocks;
}
```

```
when boat.IsActive BECOMES true
{
        string current_weather = DetermineWeather();
        SetWeatherSigns(current_weather, day.game_clock.GetCurrentTime());

        float global_difficulty = GenerateGlobalDifficulty(current_weather,
day.game_clock.GetCurrentTime(), toolbox.difficulty);
        TerrainObject OceanRocks = GenerateTerrain(toolbox.boat_position, global_difficulty);
}

while boat.IsActive
{
        wait(15 minutes); # In-game time
        boat.position += RNG.GenerateNumber(signed float, -2, 2);
        if boat.position in OceanRocks
        {
                boat.Destroy();
                player.LoseGame();
        }
}
```

4. Acceptance Tests _______9

Several features are reliant on Random Number Generation (RNG). It’s important to test the system to ensure the RNG is consistent:

Weather Determination: Weather values will be calculated 10,000 times. The expected spread should be:

- 1,000 stormy
- 2,000 rainy
- 3,000 cloudy
- 4,000 clear

Global Difficulty: Global difficulty is calculated using a randomly generated value. This value is expected to be 0.125 given the following expected value equation:

$$E[X] = \frac{0 + 0.25}{2} = \frac{0.25}{2} = 0.125$$

This value will be generated 10,000 times and will then be averaged. The result must be within one one-hundredth of the expected value.

Rock Locations: The locations of rocks will be generated based on global difficulty, but will be largely random. The boat lays on a two-dimensional Cartesian coordinate plane where -20 <= x <= 20 and -20 <= y <= 20. If the boat hits the rocks, the boat is destroyed.

There should be an even spread of rocks on the coordinate plane. A total of 100,000 rocks will be generated and the spread will be checked to ensure that it is roughly even.

Boat Position: The position of the boat randomly changes (drifts) while the player is at sea. The amount by which it drifts must be checked to ensure fairness. Due to its signed nature, the value by which the boat drifts should average out to approximately 0. A total of 10,000 values will be generated and the average calculated.

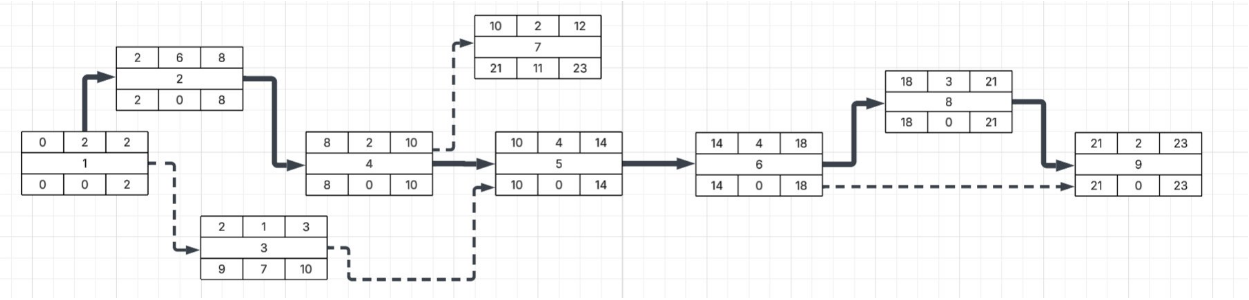
5. Timeline ________/10

Example:

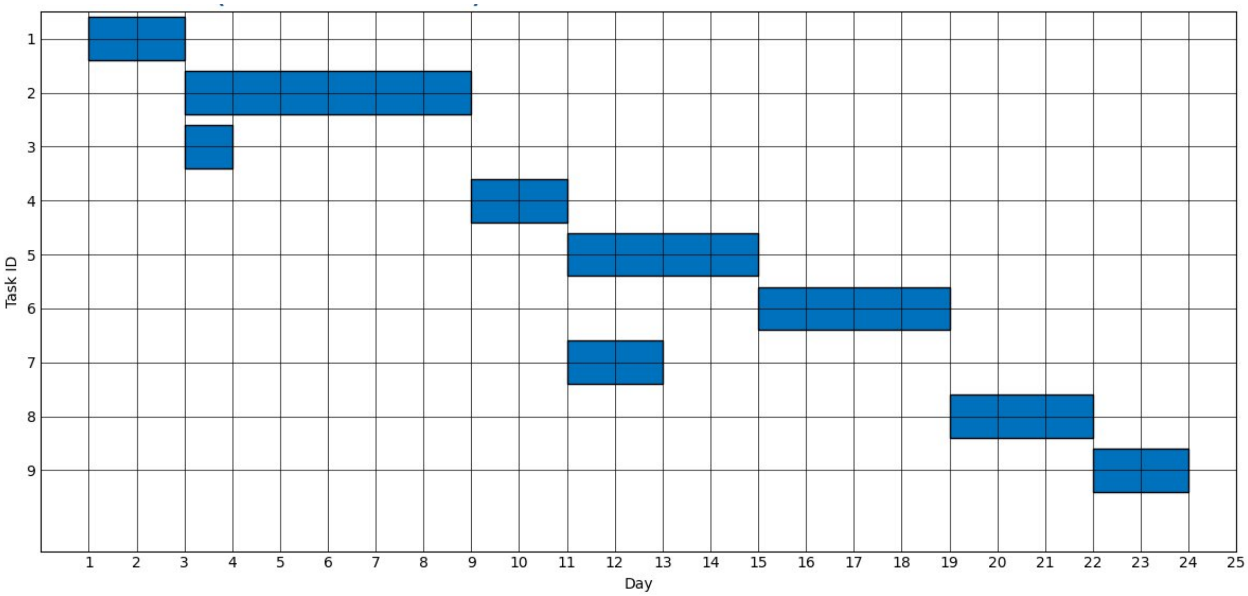
Work items

Task	Duration (Days)	Predecessor Task(s)
1. Requirements Collection	2	-
2. Difficulty Offset Calculation	6	1
3. Weather Elements Design	1	1
4. Background Elements Design and Implementation	2	2
5. RNG Programming	4	4,3
6. Integration and Balancing	4	5
7. User Documentation	2	4
8. Testing	3	6
9. Installation	2	6, 8

Pert diagram



Gantt timeline


Task ID
Day