Game Development Fundamentals Assignment 3 Rough Work

Overall Concept:

Build on the Day / night cycle code by creating a game around it.

Will create a day and night cycle based on delta.time. During the day, flowers grow on a flat plane, during the night creatures appear and will kill the player, causing game over.

The player can move back and forth (and jump if time allows) to collect the flowers before the night hits.

When it starts getting dark, the player will have to find one of 3 hiding spots: Boulder, Tree and Bush.

Adapted to two trees:

- If the player successfully hides till morning, they can start picking more flowers.
- Hiding spots are the two trees
- The player needs to get a certain number of flowers to win. (30)

Needed features:

- Day / night cycle variable with delta time.
- Encapsulation for the flowers
- Encapsulation for the player
- Encapsulation for objects to hide behind
- Collision physics for player touching flowers
- Score counter for number of flowers
- If statements for night
- If statements for win and loss

Assets:

Visual:

- Trees
- Background
- Grass
- Dirt
- Player Character
- Flowers

Audio:

- FX Day
- Music Day
- FX Night
- Walking FX
- Flower Spawn FX
- Flower Pick FX
- Death FX

Day / night cycle variable with delta time:

Need to find a way to create an oscillating value for X starting at 12 to create a variable for the program's relative time (Day / Night).

Chat GPT Solution:

The equation that matches your requirements is:

$x=6 \cdot [1-\cos(\pi y12)]$ $x = 6 \cdot [1 - \cos(\frac{\pi y}{12})]$ $x=6 \cdot [1-\cos(12\pi y)]$

Explanation:

1. The cosine function oscillates between -1 and 1.
2. Scaling by 6 gives an oscillation between -6 and 6.
3. Adding 6 shifts the range to 0 to 12.
4. The period of the cosine function is 242424, matching your pattern.

Let's check the key points:

- When $y=0y = 0y=0$, $x=6 \cdot [1-\cos(0)]=6 \cdot [1-1]=0x = 6 \cdot [1 - \cos(0)] = 6 \cdot [1 - 1] = 0x=6 \cdot [1-\cos(0)]=6 \cdot [1-1]=0$ ✅
- When $y=12y = 12y=12$, $x=6 \cdot [1-\cos(\pi)]=6 \cdot [1-(-1)]=12x = 6 \cdot [1 - \cos(\pi)] = 6 \cdot [1 - (-1)] = 12x=6 \cdot [1-\cos(\pi)]=6 \cdot [1-(-1)]=12$ ✅
- When $y=24y = 24y=24$, $x=6 \cdot [1-\cos(2\pi)]=6 \cdot [1-1]=0x = 6 \cdot [1 - \cos(2\pi)] = 6 \cdot [1 - 1] = 0x=6 \cdot [1-\cos(2\pi)]=6 \cdot [1-1]=0$ ✅
- When $y=36y = 36y=36$, $x=6 \cdot [1-\cos(3\pi)]=6 \cdot [1-(-1)]=12x = 6 \cdot [1 - \cos(3\pi)] = 6 \cdot [1 - (-1)] = 12x=6 \cdot [1-\cos(3\pi)]=6 \cdot [1-(-1)]=12$ ✅

Using Chat GPT's equation in C# for an oscillating X value:

```
float x = 6 * (1 + (float)Math.Cos(Math.PI * y / 12));
```

Allows me to declare '_timeofday' as x.

Encapsulation for the flowers:

Need to encapsulate the flowers to use as a variable item.

Encapsulation is proving complicated, instead using only arrays.

Arrays keep refreshing every iteration of the loop.

Chat GPT solution to make flowers spawn:

`Time.DeltaTime` gives us the amount of time that has passed since the last frame update. We accumulate this time in the variable `flowerSpawnTime` to keep track of the total time since the last spawn.

Every frame, this line adds the delta time to `flowerSpawnTime`, gradually increasing the value. When `flowerSpawnTime` becomes greater than or equal to our chosen interval (`flowerSpawnInterval`), it means it's time to spawn a new flower.

Each element in the array represents the x-coordinate of one flower.
We needed a way to mark whether a flower had been spawned or not, so we used `-1` as a special value to indicate an empty slot.

To make the flowers stay on the screen after spawning, we needed to:

1. Spawn new flowers only when the timer reaches the interval.
2. Draw all existing flowers on every frame update.

Solution works and is mostly understood. Need to review fully to understand.

Encapsulation for the player:

Scrapped. Doesn't need it.

Encapsulation for objects to hide behind:

Use Math.Abs for simple collision of the player model with safe zone.

Collision physics for player touching flowers:

Use Math.Abs for simple collision.

Score counter for number of flowers:

Use ++ to add score based on flower collisions.

If statements for night:

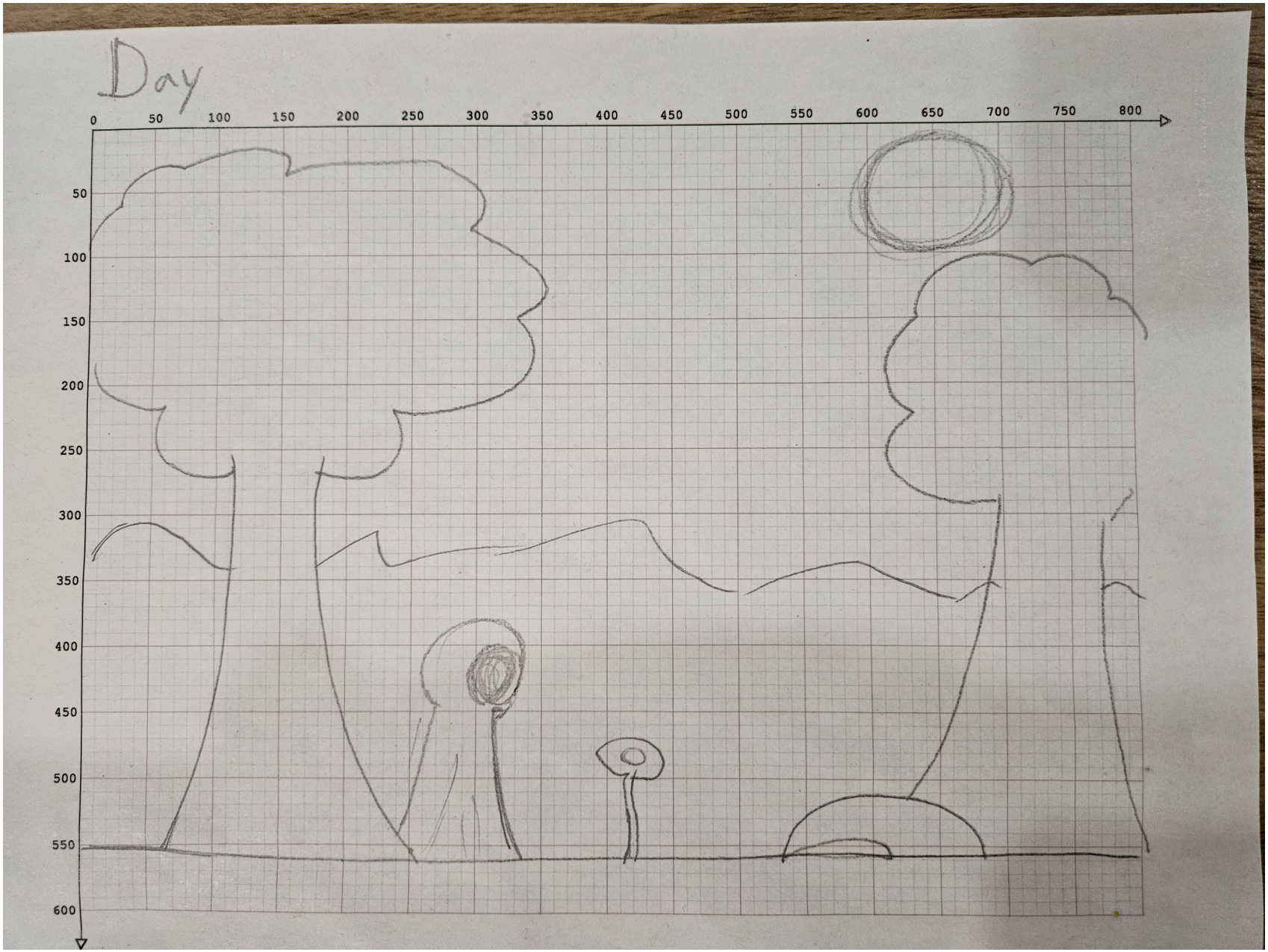
If statement for if player is not in the safe zone

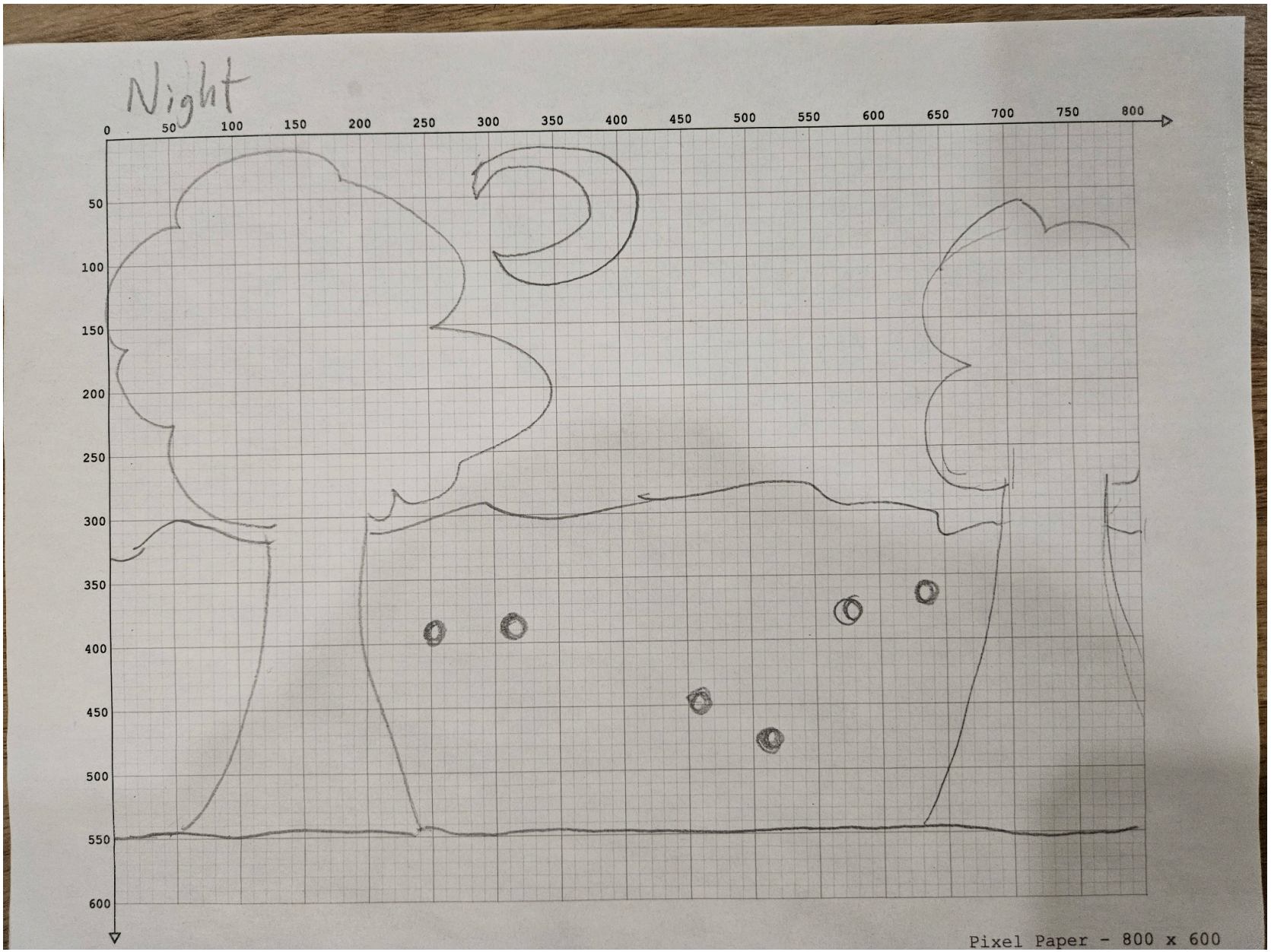
- Results in Death screen

If statements for monsters appearing at night.

If statements for win:

If the player gets 30 flowers, show victory screen.


Day
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
50
100
150
200
250
300
350
400
450
500
550
600


Night
0 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800
50 100 150 200 250 300 350 400 450 500 550 600
Pixel Paper - 800 x 600