Final Project Report:
Simple Farming Simulator

Cydney Creekmur cac614 CS428

Introduction

The goal of this project is to design a simple farming simulator game using other games like Stardew Valley and Harvest Moon as inspiration. The player will be able to plant, water, harvest, and sell crops in a small area. The game will feature essential game mechanics such as:

- A character controlled by the player to move around the environment.
- A farming system where seeds can be planted and harvested when a certain amount of time passes.
- A buying/selling system where the player can earn in-game money for the crops they harvest, and purchase new seeds to be planted.

Overall, this project will demonstrate several computer graphic concepts, like transformations, animation, and rendering, and have interactive gameplay using JavaScript.

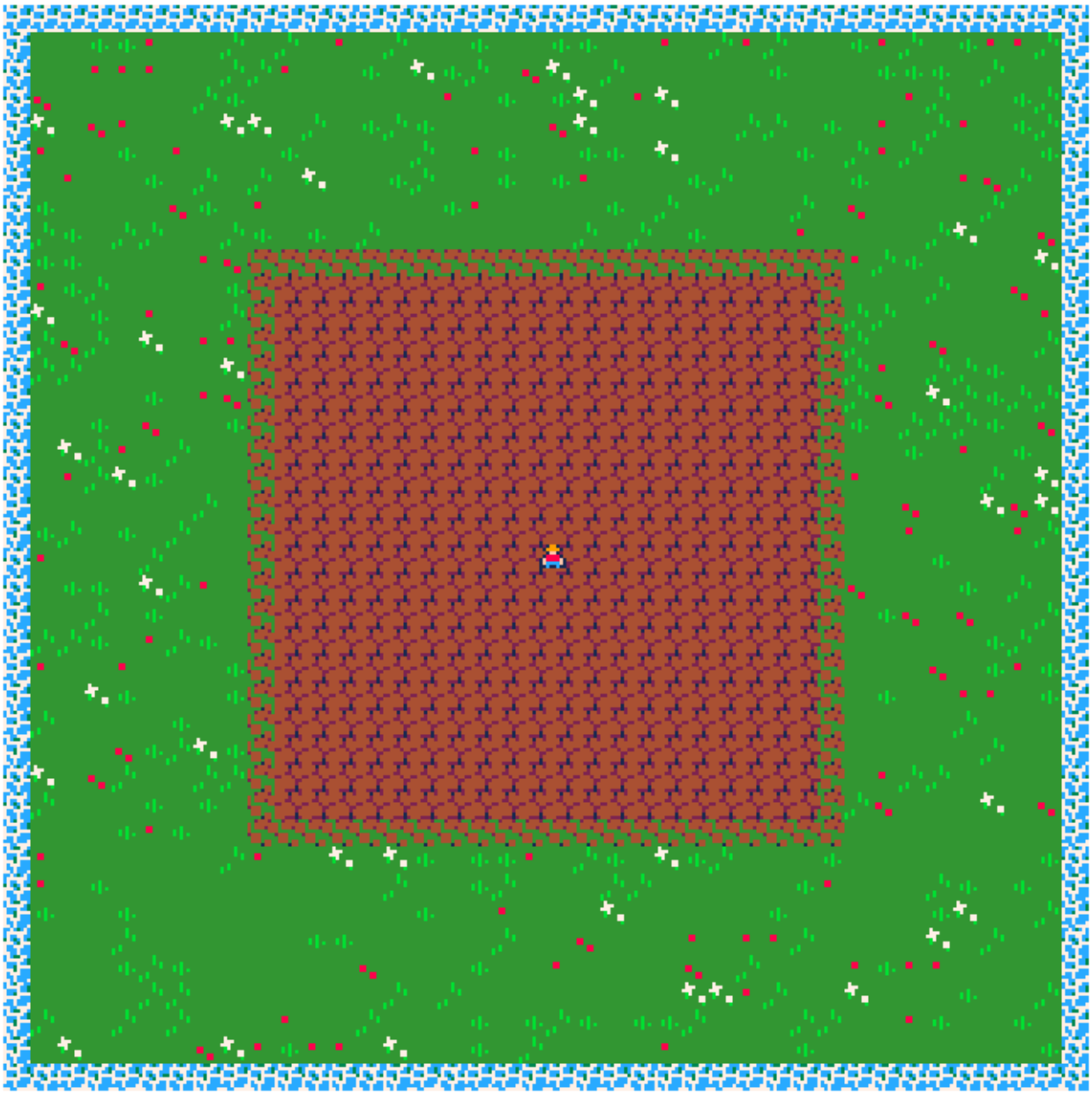
Results

Rendering and Map Generation

After initializing WebGL2 and creating shaders in GLSL, the game can render the map tile by tile in a few different sections. Map generation is handled mostly in main.js. Here, the tile types are based on location in relation to the crop field or the border of the canvas. The locations are as follows:

- Crop area → “full dirt”
- Crop border → “partial dirt”
- Outer borders → “water”
- Everything else → random ground tiles based on weights

The draw pipeline is handled in render.js which sets up buffers, texture coordinates, and draws each tile based on a scale transformation to enlarge the tiles which were previously too small. Below is a complete, generated map after restarting the browser.



Player Movement & Collision

Player movement is indicated by the keys “wasd” using keyboard listeners in input.js and movement functions in player.js. Below is an essential function in player.js that contributes to player movement.

```
    move(dx, dy) {
        const nextX = this.x + dx * this.speed;
        const nextY = this.y + dy * this.speed;

        const oldX = this.x;
```

```
        const oldY = this.y;

        this.x = nextX;
        this.y = nextY;

        if (this.checkCollision()) {
            this.x = oldX;
            this.y = oldY;
            return;
        }

        this.updateDirection(dx, dy);
    }
```

The player is also prevented from moving over water tiles at the border by checking for collisions at the map border. Below is the function created to check collisions in player.js.

```
    checkCollision() {
        const left = this.x;
        const right = this.x + PLAYER_SIZE - 1;
        const top = this.y;
        const bottom = this.y + PLAYER_SIZE - 1;

        const corners = [
            {tx: Math.floor(left / TILE_SIZE),  ty: Math.floor(top /
TILE_SIZE)},
            {tx: Math.floor(right / TILE_SIZE), ty: Math.floor(top /
TILE_SIZE)},
            {tx: Math.floor(left / TILE_SIZE),  ty: Math.floor(bottom /
TILE_SIZE)},
            {tx: Math.floor(right / TILE_SIZE), ty: Math.floor(bottom /
TILE_SIZE)}
        ];
        for(const c of corners) {
            const index = c.ty * MAP_WIDTH + c.tx;
            const tile = MAP[index];

            if(!tile) return false;

            if(tile.isMapBorder) return true; // collision
        }
```

```
        return false;
    }
```

Crop System

Planting

The player is able to plant the seeds available to them from their inventory using left click on the crop grid. Other inputs outside the crop grid result in the program becoming upset. The function to plant crops is located in player.js, and this function checks whether the:

- Tile is in the crop area
- Tile already has a crop planted
- Player has remaining seeds in their inventory
- Crop is already growing (state=1)

When a player plants a crop, the crop's state, timer, and type are all changed internally, and the tile texture updates to match the current seed type. Below is an example of the changed textures after seeds have been planted.

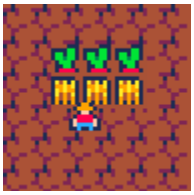
Growing

In the above image, the one at the top left has already grown. This happens because there is a growth timer inside a crop function which transitions the plant into a state (state=2) that is ready to harvest once the time threshold is met. Once grown, the texture updates at that tile as shown above. Below is the function used to handle the growing process in crop.js.

```
    updateToGrown(dt) {
        for(const crop of this.crops) {
            if(crop.state === 1) {
                crop.timer += dt;

                if(crop.timer >= this.timeUntilGrown) {
                    crop.state = 2;
                    crop.timer = 0;

                    const tileX = crop.x / TILE_SIZE + this.startX;
                    const tileY = crop.y / TILE_SIZE + this.startY;
```

```
                    const index = tileY * MAP_WIDTH + tileX;

                    if(crop.type === "radish") {
                        MAP[index].tileIndex = CROP_TILES["grown radish"];
                    } else if(crop.type === "wheat") {
                        MAP[index].tileIndex = CROP_TILES["grown wheat"];
                    }
                }
            }
        }
    }
```

Harvesting



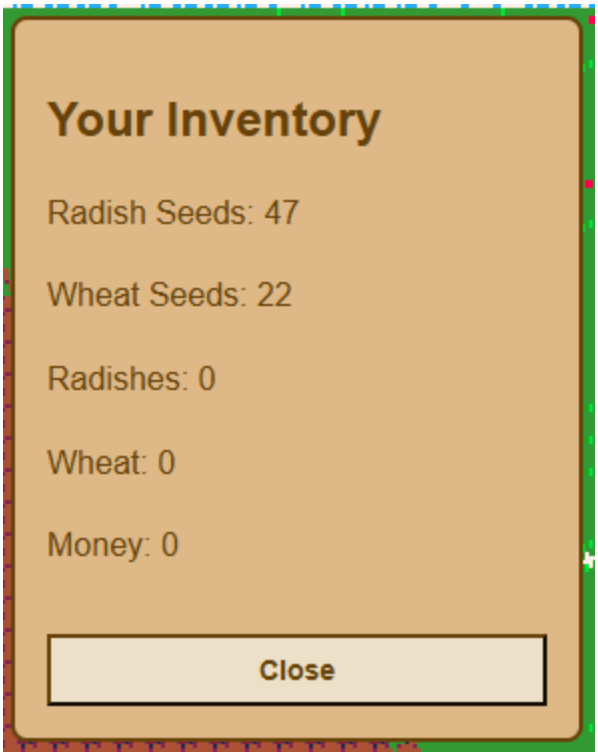
As shown above, the crops are now ready to harvest. To do so, there is a function to handle the harvesting process in player.js. After right-clicking on a grown crop, this function:

- Adds the appropriate item to the inventory
- Resets the tile to “full dirt”
- Resets the crop object’s state (state=0)

Inventory

The inventory can be displayed by pressing the inventory button, as shown below.

Once clicked, it displays the player’s seed counts, harvested crops, and money. This is done by using DOM manipulation in UI.js which is an essential class of helper functions that was created to help organize UI capabilities. Below is the inventory window shown after pressing the inventory button.



Economic System

Shop Interface

The shopping interface allows for player interaction with an in-game economic system. The shop button, once clicked, opens a functional UI that is able to:

- Buy radish and wheat seeds
- Do automatic cost calculation based on player funds by pressing the “max” button
- Show error messages fo invalid inputs

All calculations happen in UI.js using a helper function shown below.

```
function updateMaxAndTotal(player) {
    const seedType = document.getElementById("seed-type").value;
    const amount = document.getElementById("seed-amount");
    const totalCost = document.getElementById("total-cost");
    const errorBox = document.getElementById("shop-error");

    const prices = {
        radish: 2,
        wheat: 4
    };
    const price = prices[seedType];
```

```
    let quantity = parseInt(amount.value);

    if(isNaN(quantity) || quantity <= 0) {
        quantity = 1;
    }
    const max = Math.floor(player.money / price);

    if(quantity > max) {
        quantity = max;
        amount.value = quantity;
        errorBox.textContent = (`The most you can afford is ${quantity}`);
    } else {
        errorBox.textContent = "";
    }
    const cost = quantity * price;
    console.log(cost);
    totalCost.textContent = `Total: $${cost}`;
}
```

Selling Crops

All crops can be sold by pressing the “Sell All Crops” button.

The money counter and inventory values are updated after crops are sold, and a message appears above the canvas letting you know how much money you received from selling your items. Wheat and radishes are also sold at different prices. Below is the code snippet used in main.js that uses an event listener to link the button to the functionalities of the button.

```
    document.getElementById("sell-all-crops").addEventListener("click", ()
=> {
        const radishEarnings = (game.player.inventory.radishes || 0) * 5;
        const wheatEarnings = (game.player.inventory.wheat || 0) * 10;

        const earnings = radishEarnings + wheatEarnings;

        if(earnings <= 0) {
            showMessage("No crops, no money! >:(");
            return;
        }
```

```
        game.player.money += earnings;

        game.player.inventory.radishes = 0;
        game.player.inventory.wheat = 0;

        updateMoneyCounter(game.player);
        showMessage(`Sold all crops for $${earnings}.`);
    });
```

User Interface

There are multiple UI elements in this code including:

- A side menu for navigating to essential information and functions
- Buttons that open windows:
    - “Inventory”
    - “Shop”
    - “Controls” (shows controls for player movement and activity)
- Buttons that complete actions
    - “Sell All Crops”
    - Radish and wheat seeds count (toggling between the two switches the seed you are able to plant and highlights the seed you are able to plant with currently)
- Error messages & floating notifications

These interfaces make the game more interactive and intuitive. To the right is the side menu after a browser refresh.

Gameplay Performance & Stability

After playtesting this game multiple times, I have found no real issues with the performance, as it remained stable through numerous tests. Rendering utilizes WEBGL2 buffer reuse and VAOs, states update cleanly in the render loop, and crop growth and tile updates are running in O(n) time, which is small enough for the type and size of the game. Overall, there are no major frame rate issues or input delays while playtesting.

Conclusion

This project successfully met the goals described in the introduction, utilizing simple mechanics using games like Stardew Valley and Harvest Moon as inspiration. The final application is an interactive farming simulator where the player can move around the environment, plant/harvest crops, and buy and sell seeds in an in-game economy. All of these systems are combined into one loop that is responsive and visually clear for effective gameplay. Additionally, this project demonstrates an understanding of computer graphics topics that we have discussed in class, including but not limited to shaders, textures, transformations, and WebGL initialization. Overall, this farming simulator meets the goals set at the beginning of this semester, and provides a fully playable game while demonstrating ideas and principles discussed in this course.

License for Spritesheet Use

https://craftpix.net/file-licenses/