TEAM AVOCADOS

Senior Design II

Final Game Design Document

Done by group #4:

Masuda Farehia, Hammad Saeed, Amgad Morgan, Ahsan Fayyaz

A. GAME OVERVIEW

I. Project Description

- This game design document describes the details of a game inspired by Mega Man 1 on the Nintendo NES.
- It will be a 2D retro throwback with an original story and plotline, involving puzzling out how to traverse a landscape crawling with monsters, defeating bosses, meeting point quotas.
- We will also attempt to modernize the Nintendo NES gameplay mechanics to make the game overall more playable, sensible, and enjoyable – as NES games tended to be very difficult.

II. Premise

- Computer programmers and billionaires Dr. Finnegan Lockley and Dr. Lawry Wilson work together to experimentally cross AI robots and humans, called super-humans, at their company Pluto Limited.
- Dr. Lockley, who has been high-key evil this whole time, reprogrammed some of the superhumans to cause mass destruction and help him take over the world.
- Original Character (OC), a lab assistant, volunteers to be turned into a superhuman by Dr. Wilson to defeat Dr. Lockley and save the world.

III. Characters

- Dr. Fineegan Lockley (Antagonist)
- Dr. Lawry Wilson
- Megaman (Player/Hero)
- Monsters
    1. Sniper Joe
    2. Kamadoma
    3. Adhering Suzy
    4. Screwdriver
    5. Pepe
    6. BombBomb
    7. Blaster
    8. Gaboyali
    9. Blaster
    10. KillerBomb

IV. Landscapes / Levels

- Landscape 1 – Training Grounds
- Landscape 2 - Dr. Fineegan Lockley Maze

V. Story Progression & Challenge

- The player is first presented an easy training level to learn/get used to controls as well as get a feel for game mechanics + landscape.
- The game controls are:
    - Use the left/right arrows to move left/right.
    - Use the up/down arrows to climb up/down the ladders.
    - Use key C to shoot bullets.
    - Use key Enter to pause the game.
    - For developers only: use key I to become invincible.
- The second level will be much harder, will a complicated landscape and difficult monsters be crawling the pathway to the final showdown with the antagonist.
    - Player will have three lives and a health bar.
- The hero gets points for each monster that he kills, plus some monsters then die they drop bonus score points which will be added to total score points of the player after he/she defeat the Antagonist at the end.

VI. Game Play

- User Skills
  - Be able to use control buttons
  - Strategize
  - Maze Traversing
  - Stopping to thing and observe, not rushing in

- Game Mechanics (need to map these to keys)
  - Jump
  - Move Left
  - Move Right
  - Climb up/down
  - Shoot / Attack
    - Can shoot while climbing and jumping

- Items and Power Ups
  - Defeated monsters drop health food, bonus balls, and extra lifes.
  - Health power ups will be placed around the landscape

- Losing
  - OC has 3 lives.
  - OC has a heath bar - when the health bar reaches zero OC loses one life and that level and must start over / or at a checkpoint
  - Monster attack + Superhuman attacks + environmental dangers all will decrease life points

- Winning
  - If the player passes the training grounds, he processed to the antagonist Level.
  - Player defeats the boss and collect the point quota MegaMan beats the game.

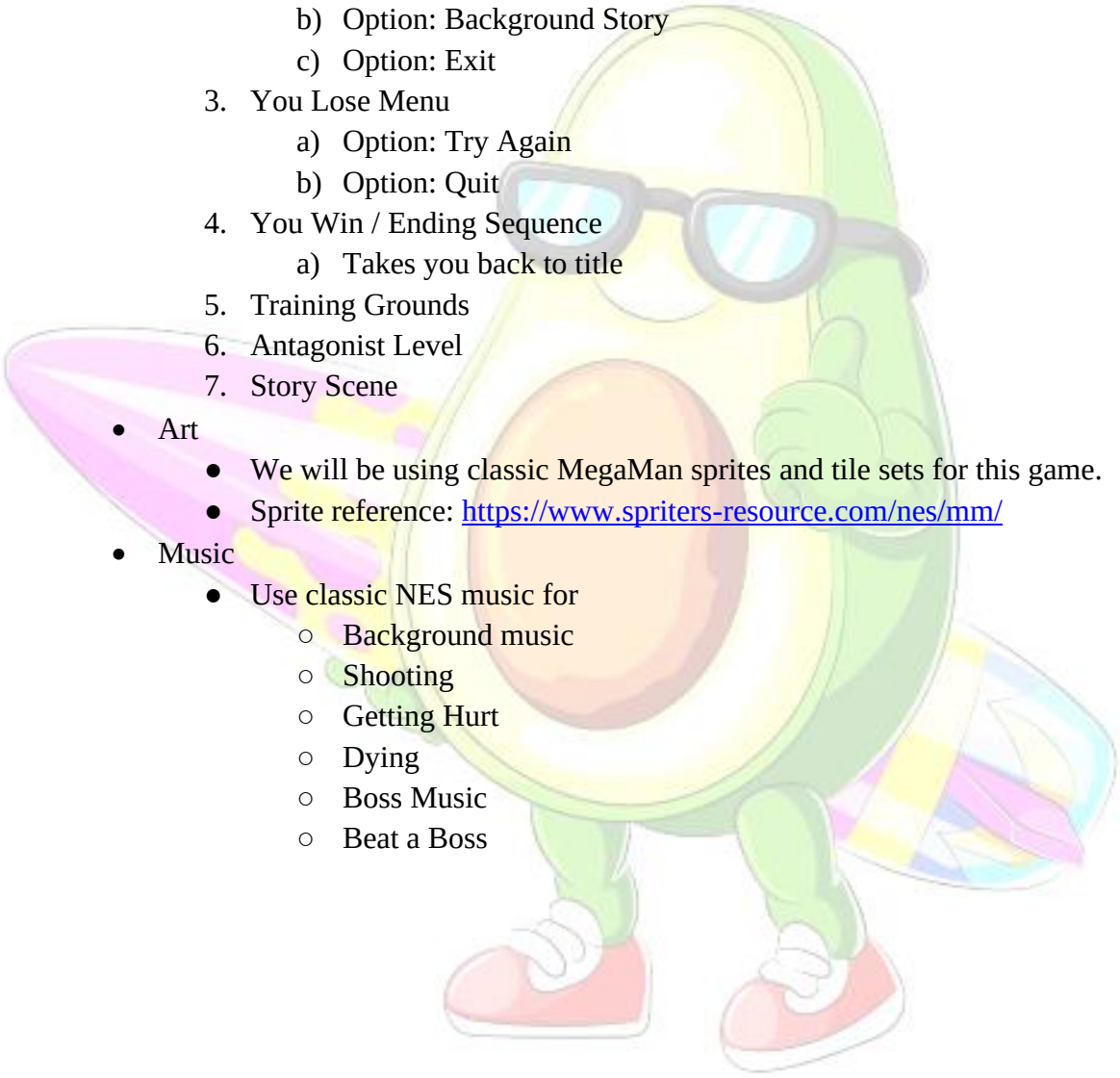
VII. Cinematics

- We are looking to maintain the atmosphere and aesthetic of Nintendo NES games. To do so we will make the following to create an NES digital feel gameplay:
  1. Sliding Window Camera Following Player
  2. Custom Transitions
  3. Black strips on edges of screen

VIII. Technical

Using UNITY 2D - Version 2019.3.10f1

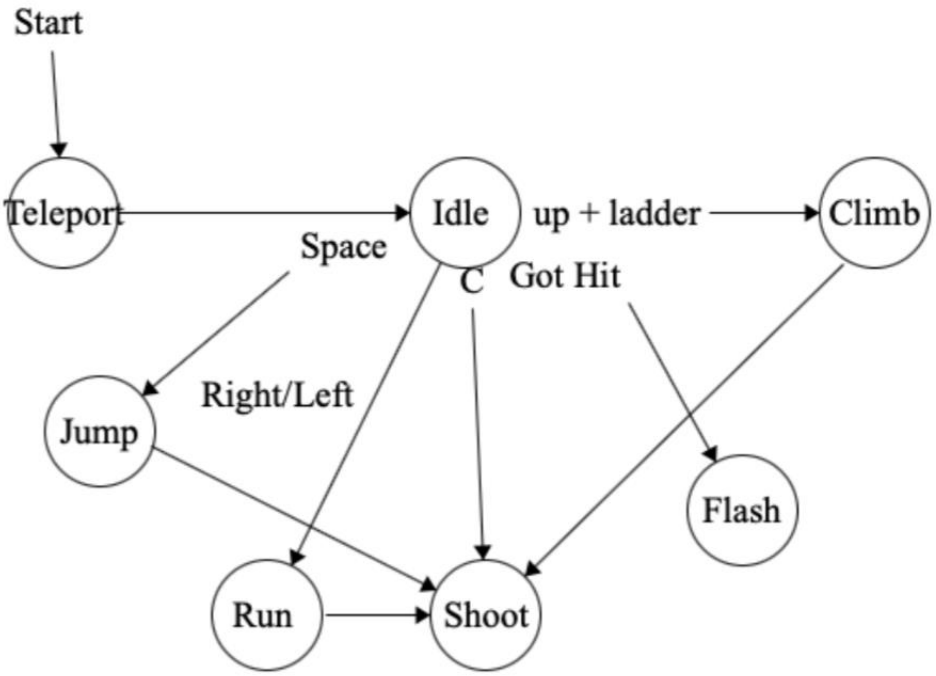
- Game Tree Map
    1. Brought to you by team Avocadoes
    2. Start Menu
        a) Option: Start Game
        b) Option: Background Story
        c) Option: Exit
    3. You Lose Menu
        a) Option: Try Again
        b) Option: Quit
    4. You Win / Ending Sequence
        a) Takes you back to title
    5. Training Grounds
    6. Antagonist Level
    7. Story Scene
- Art
    - We will be using classic MegaMan sprites and tile sets for this game.
    - Sprite reference: https://www.spriters-resource.com/nes/mm/
- Music
    - Use classic NES music for
        - Background music
        - Shooting
        - Getting Hurt
        - Dying
        - Boss Music
        - Beat a Boss



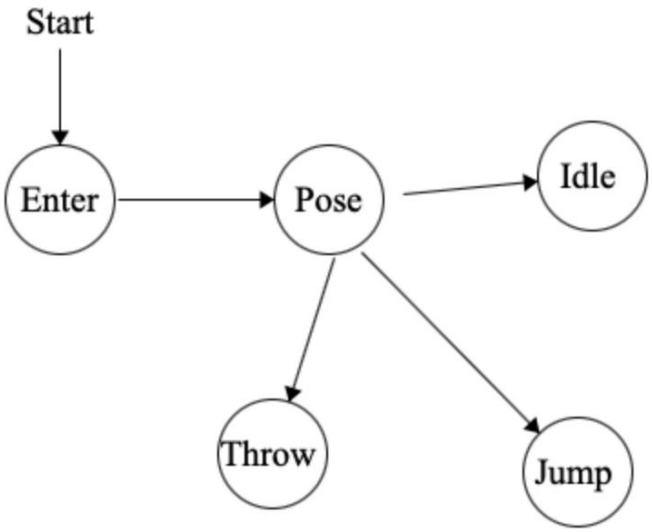
B. GAME DETAILS

State Machines

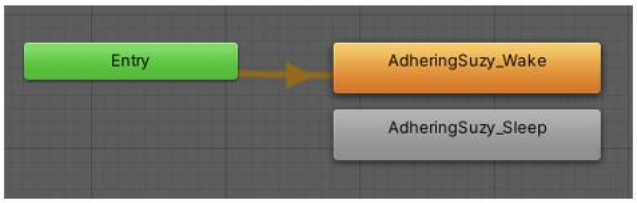
State Machine of Mega Man Movement:

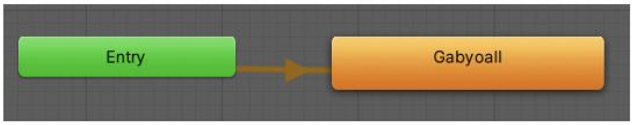


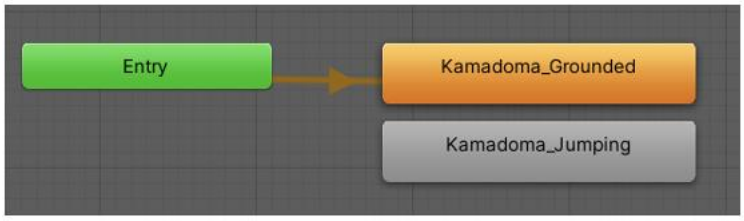
State Machine of antagonist Movement:

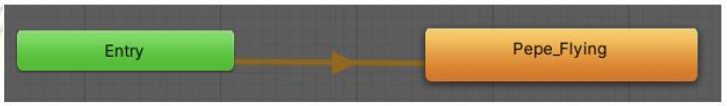


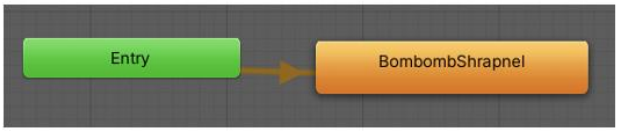
Enemy State Machines:

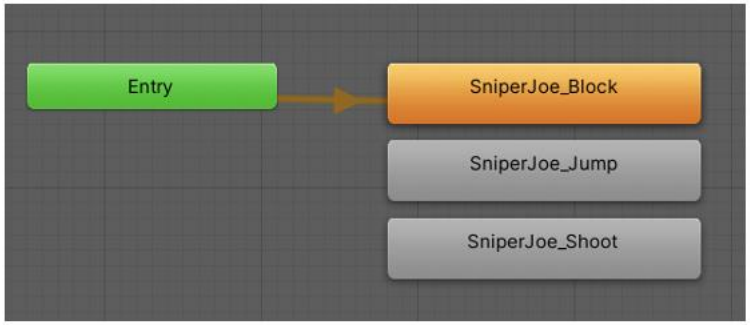


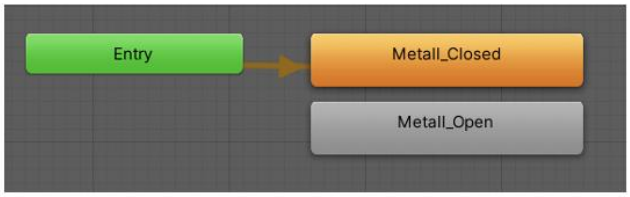


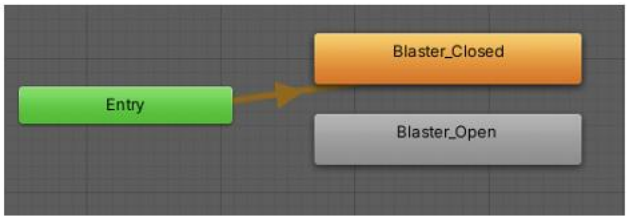


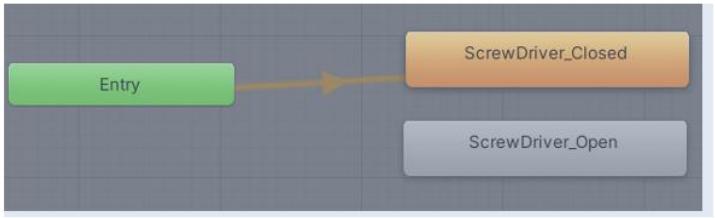


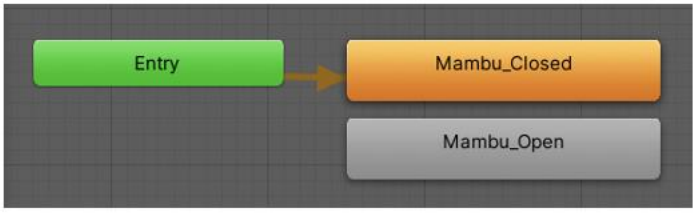


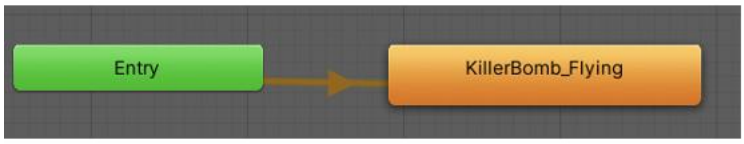












Maps:

Training Grounds:

Bomb Man Level:

C. GAME Scenes:

Start Scene:

- To familiarize the player with our team.
- Once the player press any key he/she will transfer to the Menu Scene.

Menus Scene:

- The Menu scene contain:
  - Game Title
  - Choose the play style:
    - Watch the cinematic story of the game which will lead the player after to the training grounds and then to the game level.
    - Skip the story and the training and jump into playing the game.
    - Intro scene recommended

- This script shows the arrow scene sections, Update () is called at 1fps, this takes input from the keyboard, the arrow index is updates in the array arrow Positions.

```
// Update is called once per frame
void Update()
{
    // keyboard input up arrow
    if (Input.GetKeyDown(KeyCode.UpArrow))
    {
        // cycle around to the end
        if (--arrowIndex < 0)
        {
            arrowIndex = arrowPositions.Length - 1;
        }
        // update the arrow position
        arrowSelector.localPosition = arrowPositions[arrowIndex];
    }

    // keyboard input down arrow
    if (Input.GetKeyDown(KeyCode.DownArrow))
    {
        // cycle around to the beginning
        if (++arrowIndex > arrowPositions.Length - 1)
        {
            arrowIndex = 0;
        }
        // update the arrow position
        arrowSelector.localPosition = arrowPositions[arrowIndex];
    }
```

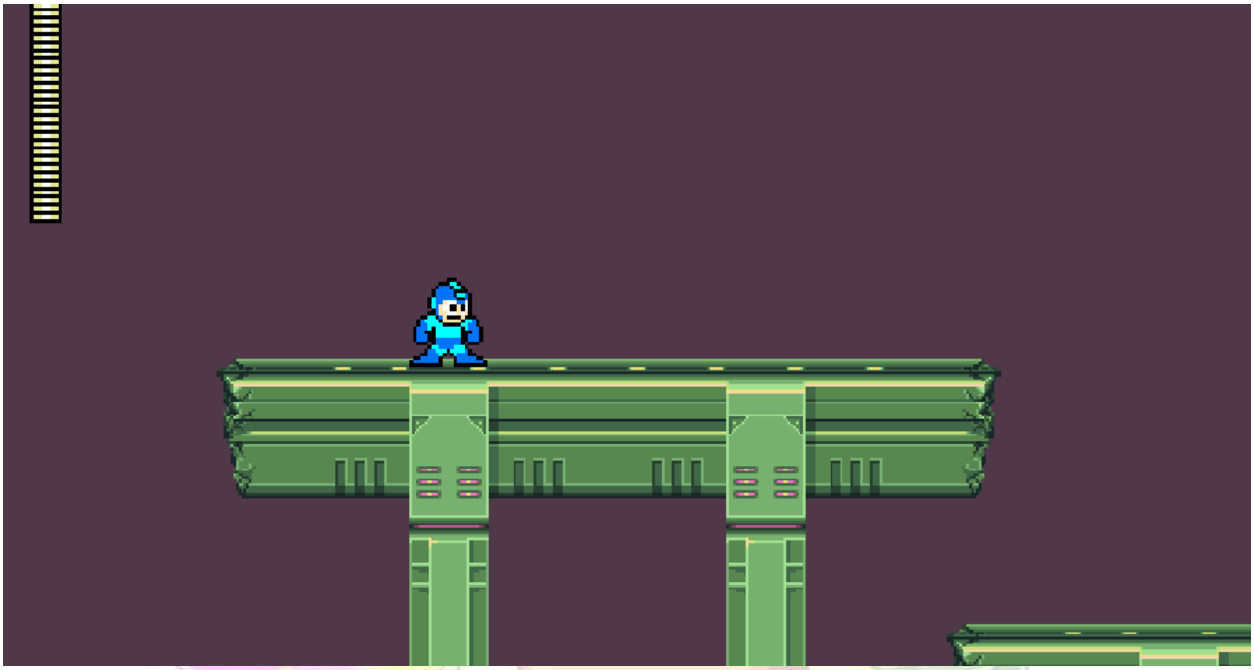
- When the enter key is hit, the scene selections occur. This is done by the arrow index, which maps the choice to the GameManager.StartNextScene function.
- This code is in the scene script for the main menu scene. Each input is registered and leads to the sound-manager giving a sound effect.

```
if (Input.GetKeyDown(KeyCode.Return))
{
    // tell GameManager to trigger the next scene
    if (!calledNextScene)
    {
        calledNextScene = true;
        // reset player numbers and energies
        GameManager.Instance.ResetPlayerLives();
        GameManager.Instance.ResetPointsCollected(false, true);
        GameManager.Instance.FillWeaponEnergies();
        // what is the arrow pointing to?
        switch (arrowIndex)
        {
            // go to the intro scene.
            case 0:
                // no parameter replay the game.
                GameManager.Instance.StartNextScene(GameManager.GameScenes.IntroScene);
                break;
            // skip to gameplay.
            case 1:
                // call the Menu scene.
                GameManager.Instance.StartNextScene(GameManager.GameScenes.BombManStage);
                break;
            // quit the game.
            case 2:
                // call the Menu scene.
                GameManager.Instance.QuitGame();
                break;
        }
    }
}
```

- Case 0 leads to the intro scene which contains dialogue, this not recommended because we took the time to create that.
- The other cases lead to the other scenes shown in the comments, the Game Manager handles that

Level Training Scene:

- Health Status is shown
- Mega Man on First Level of X
- Goal is to make it to the end without dying



```
// scene dialogue strings
string[] InstructionsdialogueStrings = {
    "DR. WILSON:\n\n\tMEGA MAN. I CAN'T STAY HERE LONG.",
    "DR. WILSON:\n\n\tMY HOLOGRAM IS VERY UNSTABLE.",
    "DR. WILSON:\n\n\tLET US BEGIN YOUR TRAINING.",
    "DR. WILSON:\n\n\tUSE THE ARROWS (LEFT, RIGHT, UP, DOWN) TO MOVE,",
    "DR. WILSON:\n\n\tUSE SPACE TO JUMP.",
    "DR. WILSON:\n\n\tSPIKES AND FALLING DOWN WILL KILL ONE OF YOUR 3 LIFES.",
    "MEGA MAN:\n\n\tTHANK YOU DR. WILSON!",
    "DR. WILSON:\n\n\tADVANCE FORWARD AND I'LL CONTACT YOU AGAIN SOON."
};

// scene dialogue strings
string[] dialogueStrings = {
    "DR. WILSON:\n\n\tMEGA MAN. TIME FOR FIGHT TRAINING.",
    "DR. WILSON:\n\n\tI ACTIVATED YOUR WEAPON.",
    "DR. WILSON:\n\n\tUSE C TO SHOOT THE ENEMIES.",
    "MEGA MAN:\n\n\tTHANK YOU DR. WILSON! YOU ARE AMAZING!",
    "DR. WILSON:\n\n\tBE CAREFULL AND FIGHT WELL."
};
```

- The dialogues rendered at the beginning of the scene can are shown if you choose to have the story play-through, you could also skip these by pressing the space key

- The timings in the dialogue are setup with time delays when shown to the screen. Here is a snippet of the code that runs:

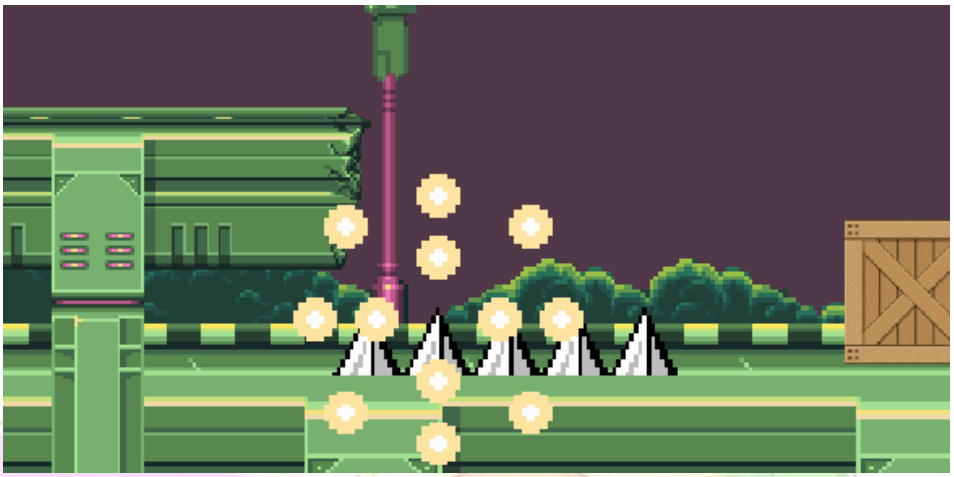
```
// dr. Wilson says he can't stay long
if (UtilityFunctions.InTime(runTime, 4.0f))
{
    dialogueBox.SetActive(true);
    dialogueText.text = InstructionsdialogueStrings[0];
}

// dr. Wilson says his hologram is unstable
if (UtilityFunctions.InTime(runTime, 7.5f))
{
    dialogueText.text = InstructionsdialogueStrings[1];
}

// dr. Wilson says let's begin the training.
if (UtilityFunctions.InTime(runTime, 10.5f))
{
    dialogueText.text = InstructionsdialogueStrings[2];
}

// dr. Wilson says use the arrows to move.
if (UtilityFunctions.InTime(runTime, 13.5f))
{
    dialogueText.text = InstructionsdialogueStrings[3];
}
```

```
// flicker out dr. Wilson hologram
if (UtilityFunctions.InTime(runTime, 25.0f))
{
    StartCoroutine(FlickerOutHologram());
}
```

- Once the player has heard Wilson's the flickering co-routing is called and the main scene begins with Mega man ending up in the new level.
- The Main Scene Script also handled the enemies of this level. The Pepe's which are the bosses on this level are created as game objects and are triggered when a checkpoint is reached.

Level Death Scene/Animation:

- Particle death scene
- Occurs on Collison with Game Objects with kill script (bombs + spikes)
- The game will allow you to restart in the even that you are killed.



```
private IEnumerator StartDefeatAnimation(bool explode)
{
    // half second delay before we do it
    yield return new WaitForSeconds(0.5f);
    // freeze player and input and go KABOOM! (if enabled)
    FreezeInput(true);
    FreezePlayer(true);
    if (explode)
    {
        GameObject explodeEffect = Instantiate(explodeEffectPrefab);
        explodeEffect.name = explodeEffectPrefab.name;
        explodeEffect.transform.position = sprite.bounds.center;
        explodeEffect.GetComponent<ExplosionScript>().SetDestroyDelay(5f);
    }
    SoundManager.Instance.Play(explodeEffectClip);
    Destroy(gameObject);
}

void StopDefeatAnimation()
{
    FreezeInput(false);
    FreezePlayer(false);
}

public void Defeat(bool explode = true)
{
    // tell the game manager we died so it can take control
    GameManager.Instance.PlayerDefeated();
    // we died! player defeat animation
    StartCoroutine(StartDefeatAnimation(explode));
}
```

- The script that controls the characters death animation. The particles are shown here. Mega Man is frozen, the explode prefab is loaded and the sound manager plays the sound.

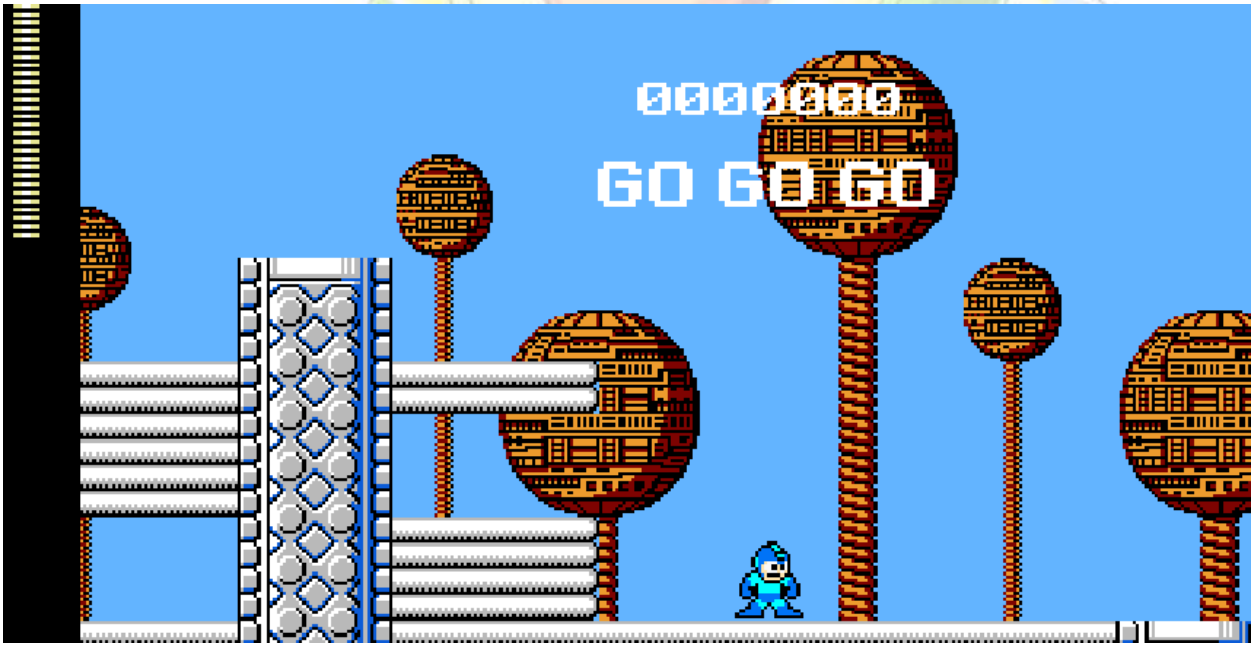
- Checkpoint code showing the collision that occurs, in the event the character dies, the respawn happens at that position.

```
private void OnTriggerEnter2D(Collider2D other)
{
    // handle this only if the checkpoint type is trigger
    //   and we only want it to trigger once
    if (checkpointType == CheckpointType.Trigger && !checkpointReached)
    {
        // the player can trigger the checkpoint being reached
        if (other.gameObject.CompareTag("Player"))
        {
            // checkpoint reached
            CheckpointReached();
        }
    }
}
```

- There is no visual representation of where the checkpoints are, the player will have to progress as far as they can without having to die. The levels are challenging.

Level Dr. Fineegan Lockley Scene:

- Bomb Man Stage
- Health bar and Score Counter are shown
- Need to make it to the end of level to Win.



- The bomb man scene holds the score from the previous scene and has it appear on the level (stats persist)
- This Boss fight against the Bomb man prevents the player from using the pause function while the fight is in progress. This is in the BombManStage script.

```
// hit the last trigger
if (UtilityFunctions.InTime(runTime, 0.001f))
{
    // don't allow the game to be paused
    GameManager.Instance.AllowGamePause(false);
    // start the boss fight music
    SoundManager.Instance.StopMusic();
    SoundManager.Instance.MusicSource.volume = 1f;
    SoundManager.Instance.PlayMusic(bossFightClip);
    // play swap tiles animation (bg flash)
    SwapTilesAnimation();
    // show bombman's health bar
    UIEnergyBars.Instance.SetValue(UIEnergyBars.EnergyBars.EnemyHealth, 0);
    UIEnergyBars.Instance.SetImage(UIEnergyBars.EnergyBars.EnemyHealth, UIEnergyBars.EnergyBarTypes.BombMan);
    UIEnergyBars.Instance.SetVisibility(UIEnergyBars.EnergyBars.EnemyHealth, true);
    // let go of the ladder and freeze input
    player.GetComponent<PlayerController>().ResetClimbing();
    player.GetComponent<PlayerController>().FreezeInput(true);
}

// close the doorway to the battle area
if (UtilityFunctions.InTime(runTime, 3.0f))
{
    ToggleDoorway2();
}

// show bombman
if (UtilityFunctions.InTime(runTime, 4.0f))
{
    InstantiateBombMan("BombMan", objectPosition["BombMan"]);
}

// do bombman's pose and fill health bar
if (UtilityFunctions.InTime(runTime, 4.5f))
{
    bombMan.GetComponent<BombManController>().Pose();
    StartCoroutine(FillEnemyHealthBar());
}

// battle starts, enable boss ai and give player control
if (UtilityFunctions.InTime(runTime, 5.75f))
{
    bombMan.GetComponent<BombManController>().EnableAI(true);
    player.GetComponent<PlayerController>().FreezeInput(false);
    // allow the game to be paused
    GameManager.Instance.AllowGamePause(true);
    // move on to BossFight state
    levelState = LevelStates.BossFight;
}
break;
```

- Looking at the InTime function, this uses delays to create the game scene setup, form introducing the Boss and animations to givin the player control.

- Kamadoma from the original game are are shown here, they follow the characters last position and make jumps towards the character. Avoiding is reccomended.
- These enemies are controlled thorugh the KamadomaController script.

```
if (isGrounded)
{
    animator.Play("Kamadoma_Grounded");
    rb2d.velocity = new Vector2(0, rb2d.velocity.y);
    jumpTimer -= Time.deltaTime;
    if (jumpTimer < 0)
    {
        // randomly choose between the two jump vectors
        jumpVector = jumpVectors[Random.Range(0, 2)];
        if (playerPosition.x <= transform.position.x)
        {
            // player is to the left of the enemy
            jumpVector.x *= -1;
        }
        // apply jump vector and reset jump timer
        rb2d.velocity = jumpVector;
        jumpTimer = jumpDelay;
    }
}
```

- At each instance that the enemy becomes grounded, the palyers relative position is calculated and a jump is made towards the direction of Mega Man.
- The jump can be made in two version which is randomly decided by the jump vector, random.range(0,2)

Dr. Fineegan Lockley Fight:

```
[SerializeField]
Vector2[] shrapnelVectors =
{
    new Vector2(-0.75f, 2f),
    new Vector2(-1.5f, 2.5f),
    new Vector2(1.5f, 2.5f),
    new Vector2(0.75f, 2f)
};

[Header("Prefabs")]
[SerializeField] GameObject bombPrefab;
[SerializeField] GameObject shrapnelPrefab;

[Header("Audio Clips")]
[SerializeField] AudioClip explosionClip;
```

- Each bomb that is thrown by this enemy has a splash effect of damage. This is called the shrapnel from the bomb.
- The movment and directoin of the bomb thrown is also conrotlled similary to other enemies, relative to the position of where the player was last.

Game Over Scene:

- This Scene is handled with the Game Over script, the arrow selection follows very similarly to the main menu scene. The arrows are stored in an index and drawn to the screen at 1fps.
- The Score for the screen is shown if the player makes it to the end of the game. This is a challenging game; it may not be possible to get to see the score screen.

```
// keyboard input enter
if (Input.GetKeyDown(KeyCode.Return))
{
    // tell GameManager to trigger the next scene
    if (!calledNextScene)
    {
        calledNextScene = true;
        // reset player numbers and energies
        GameManager.Instance.ResetPlayerLives();
        GameManager.Instance.ResetPointsCollected(false, true);
        GameManager.Instance.FillWeaponEnergies();
        // what is the arrow pointing to?
        switch (arrowIndex)
        {
            // continue
            case 0:
                // no parameter replay the game.
                GameManager.Instance.StartNextScene(GameManager.GameScenes.BombManStage);
                break;
            // stage select
            case 1:
                // call the Menu scene.
                GameManager.Instance.StartNextScene(GameManager.GameScenes.MenuScene);
                break;
            case 2:
                // call the Menu scene.
                GameManager.Instance.QuitGame();
                break;
        }
    }
}
```