ART 4059/EE 4859 Final Report:

Adam Schwarzenbach, Austin Carter, Madison Manale,

Robert Boese, Tristan Evans

Github: https://github.com/atcarter1999/DM-Capstone

Youtube Demo: https://youtu.be/hEBflqB0IiI

The final project created by our group is a 2D action platformer most closely fitting the description of a metroidvania. The game was created in the Unity game engine, with the use of other programs including Aseprite, Autodesk Sketchbook, EarSketch, Adobe Illustrator, and Procreate. The main aspects of the game can be broken into five main components: Art, Animation, Audio, Gameplay, and Level Design.

- Art:
    - There were many 2D sketches, but ultimately the pixel art is what made it into the game. The majority of concept work for the art assets was created in Procreate, while the pixel art assets were created using Aseprite. For the sake of cohesion, we kept the colors of most of the assets confined to a simple, cool toned palette. One of our main colors used as a motif throughout the game was a jewel-toned purple. This color can be seen throughout the UI, logo, tile maps, and characters.
    - 2D art: The non pixel artworks were supposed to serve as a close up image of the NPC's face during the dialogue and UI options. In addition, we constructed a logo for our game that was later finalized in Adobe Illustrator. The logo has a hand-drawn feel with a purple gem sitting in the 'A.'

    - Pixel art: The pixel art characters and world were used to give off an old fashioned look to the game. Using the concepts drawn in procreate, Aseprite was used to draft the static sprites. For the sprites, we chose to keep the value range simple in order to not over-exert ourselves shading. The exception to this would be the protagonist, which has the largest range of values. Our tile maps were also created in Aseprite.
- Animation: Most creatures had simple animations of a few frames. The two exceptions were the main character and the boss. Each animation used in our game was created using hand drawn, frame-by-frame animation. Despite the lengthy process, frame-by-frame animation gave our artists more control over each of the animations needed for the game. For our boss and player animations, which included attacks and walk cycles, we used human movement as reference.
- Audio:
    - The audio for the game can be further split into two categories: Music and Sound Effects
    - Music: Similar to the art style, the music attempted to have an archaic element. Most of the score was designed to be played on loop for several minutes, but music like the main menu theme was designed for a one and done experience.
    - Sound Effects: Free audio assets were used with audio listener, audio source, and audio clip components in the unity engine.

- Gameplay:
    - Gameplay can be further split into four categories: Player Control, Enemy AI, Damage/Death and Object/NPC interaction
    - Player Control: The primary input method for the game is through the use of a controller, namely xbox or playstation. There exists scripting for keyboard input although it was never finished. An additional package for the Unity engine called Unity.InputSystem was added to the project to allow for controller support. There are seven actions mapped to the controller: Moving, attacking, jumping, Interacting, Healing, Dashing, and Pausing.
        - Movement: Player movement is mapped to the left stick of a controller and allows movement left and right
        - Attack: Attacking is mapped to the left button of a controller (the "X" button on an xbox controller) and is directional based on the position on the left stick. For example, if the player moved the left stick up and attacked, the attack will be over the character's head. Same for left, right, and down. Note: The player can only attack down if they are in the air. Hitting an enemy restores an amount of "essence" which is a resource the player uses for healing
        - Jump: Jumping is mapped to the bottom button on a controller (the "A" button on an xbox controller). In this game there is variable jumping, meaning the longer you hold down the jump button, the higher you jump up to a maximum height. The player has free

horizontal control while in the air. There are also two types of jumps, a normal jump as detailed above but also a wall jump. Wall jumping works almost exactly as normal jumping, except the player must be on a wall to use it.

- Interaction: The player can interact with certain objects and NPCs, this action is mapped to the top button on a controller (the “Y” button on an xbox controller)
- Heal: The player has an amount of hits they can take before death, they can also heal from any damage taken. By holding down the right button of a controller (the “B” button on an xbox controller), the player will begin to rapidly decrease their “essence” resource. Once 20 essence is used uninterrupted, one health point will be restored.
- Dash: Dashing allows the player to rapidly move a distance horizontally. This action is mapped to the right trigger of a game controller.
- Pause: Pausing allows the player to freeze in game time and access a pause menu, where the user can either unpause the game or exit the current level to the main menu. This action is mapped to the “Start” button (the small rightmost button near the center of a standard controller). Invoking the pause state freezes in game time including enemy movement, player movement, player controls (as

the menu controls are activated) and all other game objects that use in game time.

- Enemy AI: Enemy AI was designed using a State machine. Based on where the player is and other environmental factors, the AI will be put into a state that determines its actions. These actions are determined by a separate script for each enemy type which communicates with the state machine. There are there state machines in the game: Walking AI, Flying AI, and Boss AI
    - Walking: There are 3 states in the Walking AI, Patrol, Chase, and Attack. An enemy using this state machine will be in the patrol state until the player is detected. They will then move to the chase state, and move toward the player. Once in a specified range, they will move to the attack state, and try to hit the player with the unique attack of its enemy type. Player detection is achieved by a horizontal ray cast using a layer mask for the player. The ray is a specified length and does not penetrate walls. Two enemies use this state machine, the slime and the crawler.
        - Slime: The slime will hop around until detecting the player, where it will then hop in the player's direction to chase/hurt them.
        - Crawler: The crawler is a crystal insectoid that will patrol an area until detecting the player, where it will then chase

    them. Once close enough, it will do a forward attack to damage the player

- <u>Flying:</u> The flying state machine has the same states as the walking state machine, the difference lies in the player detection. Any enemy with this state machine has a circle detection radius rather than a horizontal raycast. The circle radius cannot see though two enemies use this state machine: the bat, and the flower turret.
    - Bat: Will fly around a room until seeing a player, will then chase down the player to hurt them.
    - Flower Turret: Uses the flying state machine to detect the player but does not utilize it further, Once the player is detected, it will shoot projectiles at the player.
- <u>Boss:</u> The boss state machine is unique in that player detection does not affect state transitions, as the boss and the player are locked in the same room throughout the fight. Rather, state transitions are determined by the distance of the player to the boss. There are long distance actions, mid range actions, and close range actions. There is only one boss in the project upon turn in, the golem.
    - Golem: The golem has 4 attacks. If the player is close range, it will punch and slam the ground in an area of effect. If the player is mid range, then it will shoot a spike from the ground up on the player's position or jump at the

player and do the close range ground slam. It has no long range attack and will instead prioritize getting closer to the player.

- Damage/Death: If the player touches an enemy, is hit by an enemy attack, or touches a damaging object such as spikes or falling stalactites, the player will take one damage and will become immune to future damage for 2 seconds. If the player reaches zero health they will die and spawn at the last fire they interacted with.
- Object/NPC Interaction:
  - NPCs: If the player enters a certain radius next to NPCs, they will be able to interact with them by pressing the interact Y button on their controller. This is primarily used for conversations with NPCs that are either essential or non essential to the game's completion. For example there is an NPC directly in front of the player when they spawn in, and in order to activate the first campfire checkpoint, they must interact fully with that NPC. While interacting with an NPC, the player UI has a text box located under the player that continuously updates a text dialogue, and the player must press the interact button after each dialogue.
  - Objects: Sometimes when the player interacts with an NPC, they trigger an ingame collectible object to spawn, aka a Quest, where the player must find this object and pick it up. This simply involves the player walking into the object and triggering an in

game response dependent on which object the player collects. There are 3 in-game objects to be collected, all non-essential for completion of the game but still incredibly useful. The first is a Dash Unlock orb that grants the player the ability to perform the Dash movement mechanic. The second is a purple ore called Matterium that if brought back to the blacksmith, doubles the player's damage power value. Last is a purple Health Crystal, which increases the player's max health by one.

- Level Design:
    - The overall level structure was designed with the overarching theme of exploration, platforming of incrementally increasing difficulty, and a variety of scenarios in which the player has to combat challenges with unique combinations and applications of their toolkit.
    - The setting is that of a distant, uncharted world, teeming with foreign creatures both hostile and friendly. Throughout the two main areas in the game (forest and cave), there is a unifying theme of purple crystalline matter, which acts as an element of "corruption" throughout the environment and gets more intense and noticeable as the protagonist traverses further into the world. This color/motif is a fairly common constant among the threats of the environment, signifying danger to the player at a glance.
    - Lastly, the levels were designed with Unity's built-in tilemap system, which provides an easy and accessible method of mapping out a 2D scene

by placing repeatable segments of a given spritesheet into a grid, and assigning any unifying physics/layer properties to this grid-style object. The two levels/subsections each used different tilemaps to give them unique appearances, but the physical properties of the tiles remained uniform throughout the project.

- **Forest**: intro to the game's mechanics, objectives, and themes
  - At start, introduces NPC interaction and the checkpoint mechanic (campfires)
  - Introduction to basic platforming (tree branches) and basic combat (slow, non-threatening enemy types - bat/slime)
  - NPC village - adds character to the world, introduces upgrades (dash mechanic) and sidequests with rewards (damage increase)
  - Optional sidequest path introducing slightly trickier platforming with spiked (damaging) terrain and more aggressive enemies (flower turrets)
  - Gradual transition near the end of the forest area to cave entrance, communicated through dark color gradient/tilemap shifts and more challenging platforming
- **Cave**: intermediate-level test of applying skills learned at the start
  - Introduction of more environmental hazards, forcing the player to be more deliberate with their movement abilities (narrow spiked passages, falling stalactites)

- Path to an optional upgrade (health increase) with no NPC signifying it - requires player to explore off the beaten path
- More difficult enemies that slowly but relentlessly attack player (crawlers)
- End of level culminates in a boss fight against a crystal-infused golem, whose AI forces the player to react and dodge to multiple attacks to prevail; this is a relatively difficult fight that will reward those who took the extra time to strengthen all of their abilities throughout the game

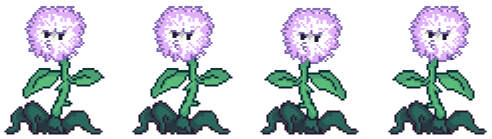
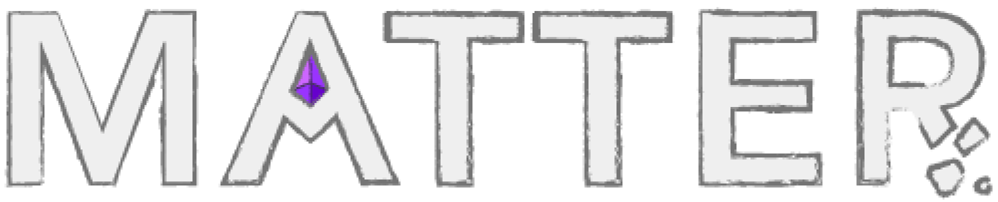
Group Members Accomplishments:

- Adam Schwarzenbach
    - Drew initial character/enemy/logo designs
    - Composed many music pieces
    - Drew up close facial character design for NPCs
    - Create a draft of the dialogue system
    - Drew UI designs for start, pause, controls, and quit
    - Animated enemies
- Tristan Evans
    - Implemented movement, dashing, and jumping
    - Implemented Player Combat and Enemy Code
    - Implemented mini boss and AI
    - Scripting
    - Debugging
    - Sound design
- Robert Boese
    - Demo gameplay
    - Created pixel art assets for enemies, environments, and boss
    - Level design
    - Audio Implementation
- Austin Carter
    - Demo gameplay
    - Pause state

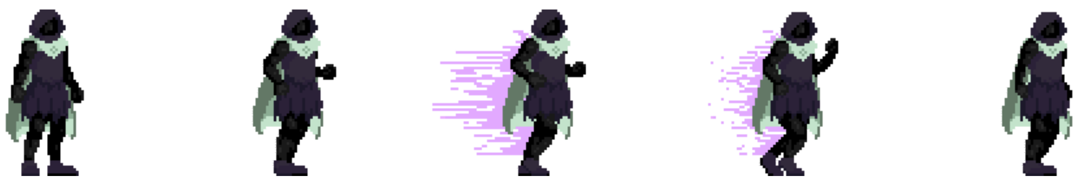
    - UI / player overlay
    - Campfire checkpoints
    - NPC interactions / quests
    - Animation implementation
- <u>Madison Manale</u>
    - Finalized logo design
    - Created pixel art for protagonist, enemies, and health gems
    - Forest and cave tile maps
    - Background artwork
    - Animated player and boss

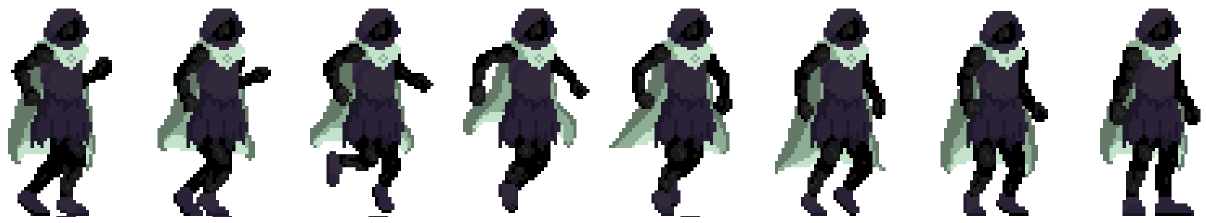
Highlights:

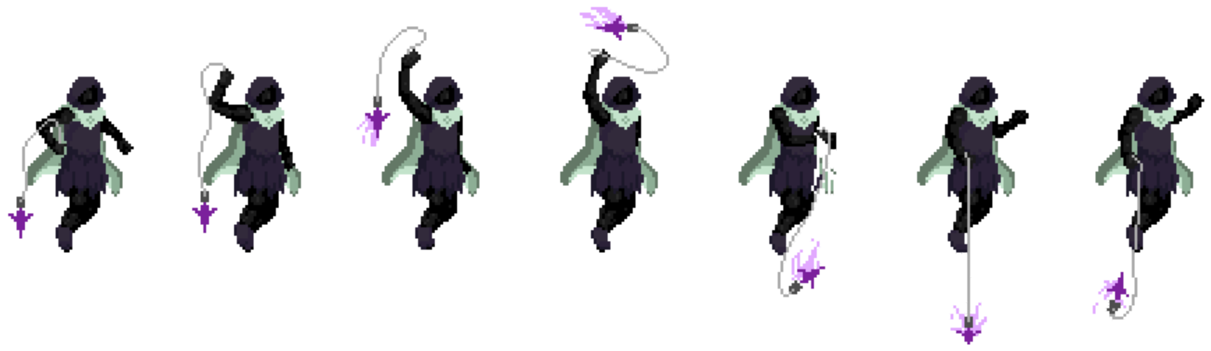
Art: Here are some of our favorite art assets created and used in our game:

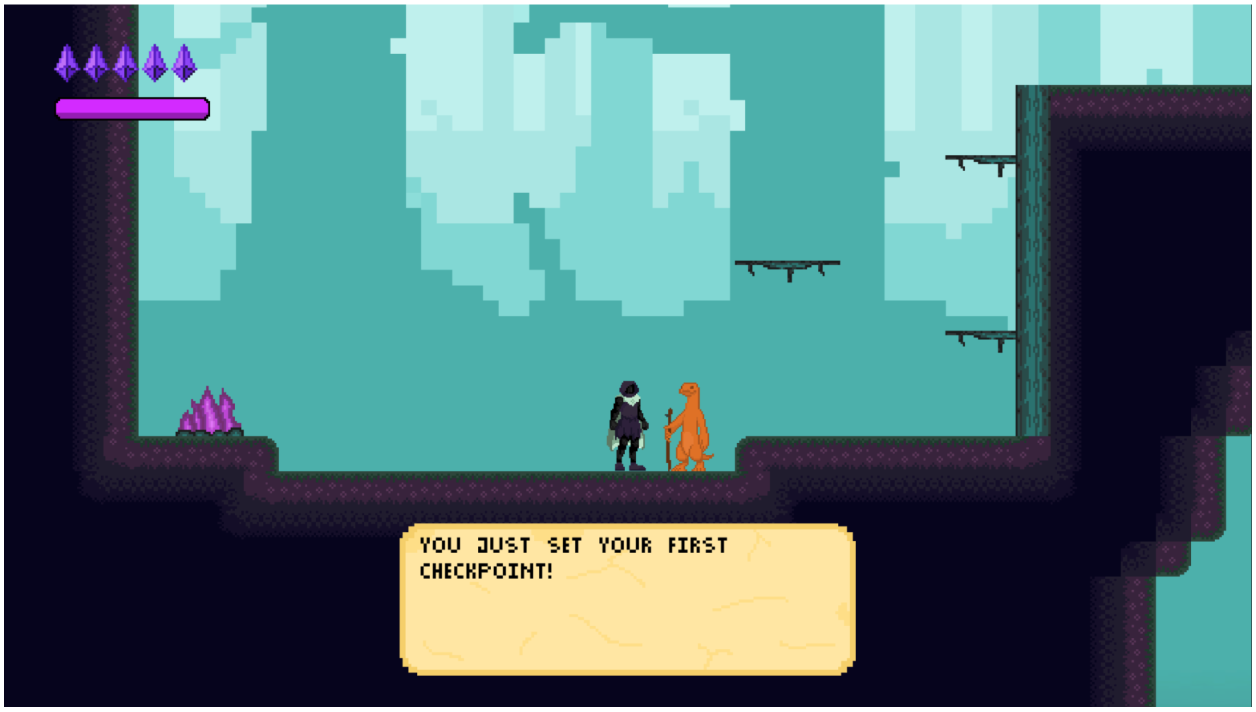
Player Mechanics/Movement: Dashing, Wall Jumping, and Directional Attack were the most advanced and favorite player mechanics that we implemented







NPC/Campfire Interactions: Specifically, the relationship between the campfires, NPCs, and the level design gave the player purpose and direction throughout the game.



Music: Here is the final boss theme of the first level. This one is not long, but it is implemented with several effects.

https://drive.google.com/file/d/1Ds0XCcvfFyceae77QTmH2bWcfh1spqLc/view?usp=sharing

Challenges and Solutions:

Individual	**Challenge**	**Solution**
Adam	Trouble figuring out how to contribute once music and still art was done	Trying out new programs such as Aseprite and Unity for animation and game design
Austin	Unity switching from Collaborate to Plastic SCM as their in house source control	Meeting as a group to verify our project was still intact, and constantly helping each other learn Plastic SCM
Robert	Trying to work with an ever evolving scale because we never settled on one scale	Experimentation through trial and error and eventually getting the right solution
Tristain	Programming physics can be difficult when dealing with niche scenarios	Admitting the lack of knowledge of physics and reviewing material to better implement it
Madison	2D Animation in a new medium	Reviewing Aseprite tutorials and creating stick-figure keyframes to build on top of

Post Mortem:

There were some features that were not implemented or just failed on presentation day. One feature was self healing. Self healing was working and showcased in a previous progress report. Additionally, the cave theme and boss theme audio assets were not played in the final demo.

On the other hand, our group regularly met every Monday and Wednesday, regardless of whether we had zoom class or not, to check in on the project and discuss further changes. This methodology was extremely effective and partially responsible for our long term success with maintaining deadlines and effectively doing our individual tasks. Also worth noting, each of our group members contributed in a substantial way to the project.