

Card Mage

Computer Science and Engineering

University of Nevada, Reno

Team 13

Robert Bothne, Grant Davis, Dan Huynhvo, Abida Mim

Instructors: Dave Fiel-Seifer, Devrin Lee

Outside Advisors: Jared Freeman (LionDragon Studio), Rob McKay (Aristocrat Games),

Michael Wilson (UNR)

March 3, 2023

Table of Contents	2
1. Abstract	3
2. Project Updates and Changes	3
3. User Stories and Acceptance Criteria	3
4. Testing Workflow	4
5. Testing Strategy	5
6. Unit Testing	6
7. Contributions of team members	8

1. Abstract

Card Mage is a video game project that combines the strategy of card games with the excitement of action-packed dungeon crawling roguelike games. The player takes on the role of a mage who uses cards to cast a wide variety of spells that are used to defeat enemies while exploring a procedurally generated dungeon. The game's unique gameplay mechanics make it an immersive and fun experience. Players can customize their playstyle with a wide variety of spell cards. With a new dungeon layout generated for each playthrough, Card Mage offers a fresh experience every time. This project is important for the entertainment it provides for its users and the creativity it can inspire.

2. Project Updates and Changes

As far as updates to the project and changes in design goals, we have explored several options. We are currently in an intermediary phase of development where we have reduced functionality for the sake of long term success. In this regard we have implemented visually derelict UI systems and have reconfigured our AI path processing to utilize a navmesh base. This has resulted in a currently weaker pathfinding algorithm that does not properly track the player in the desired manner but will ultimately provide a stronger foundation for future developments in the field of AI. In regard to our UI systems they require polish in terms of design for ease of accessibility through a cleaner and easier to read system. Our changes have been a shift of focus of what we deem important, which we have deemed to be the AI system, deck system, and a need for a plot. The deck system now allows the user to get rid of cards in the deck editor of the pause screen. We have added a stats page for the user to view multiple elements of gameplay for the user (such as time spent, currency collected, and enemy killed), and for the stats page we have made the data containment of the stats more robust and unlikely to lose data in between game runs. We plan to add more elements in the future such as accuracy of spells hit, and levels climbed.

3. User Stories and Acceptance Criteria

UI:

As a person who has eyes very sensitive to light, I want a brightness setting so that I can keep overall brightness to a comfortable level..

As a gamer who likes to think through my actions in a card game, I want a pause option so that I can stop mid game if I need to take time to rethink my strategy.

Enemy AI:

As someone who enjoys a variety of enemy difficulty in games, I want a variety of enemies so that I can feel a challenge with constant new difficulties..

As someone who enjoys interactive combat, I want enemies that react to my positioning and movement patterns.

Card System:

As a trading card collector, I want to be able to build and customize my own deck to use in game on top of collecting cards throughout the run.

As someone who plays roguelike games, I want the card system to have a feeling of luck in their mechanics.

Level Generation:

As a person who enjoys the idea of getting stronger, I want to feel a sense of progression as I traverse the rooms through powerups.

As a person who finds games challenging, I want a guide towards the end if I am unable to find it and frustrated at a lack of progress.

4. Testing Workflow

Path workflow:

1. Asset Creation Workflow: Validated at each stage by team members, and final review by other team members.
2. Bug fixing workflow: Validated at each step by team members, including reproducing and diagnosing the issues, verifying and validating each fix, and ensuring no new bugs have occurred.
3. Level design workflow: Validated at each stage by team members, and through playtesting with team members or playtesters to gather feedback.
4. Audio workflow: Validate at each step by team members, ensuring that audio matches with intended tone and mood, is being integrated correctly into the game engine.

Unhappy path workflow:

1. Error handling workflow: Validated at each stage by team members, by observing the error handling behavior and ensuring clear and helpful error messages are provided to the user.
2. Performance workflow: Validated at each stage by team members by testing the game under different performance conditions to ensure optimal performance and no crashes or other issues.
3. AI behavior workflow: Validated at each stage by team members, by testing AI under different scenarios to ensure correct behavior and no unexpected exploits or issues.
4. Level generation workflow: Validated at each stage by the team members, by testing the level generator under different scenarios to ensure varied and interesting level generation that is balanced and consistent with the game's design, and no unexpected level generation or crashes occur.

5. Testing Strategy

Our testing strategy primarily relies on unit tests performed through the course of development with team meetings to conduct acceptance tests and determine whether our goals are met and whether or not these goals reflect what we wish to achieve. Generally our methods so far have involved all users cross testing various systems in a single test environment. To this end we generally provide feedback in relation to systems in which we were minimally involved in the development of. Robert Bothne generally provides feedback about hero mechanics, AI, and UI design. Abida Mim generally provides feedback about AI, hero mechanics and level design. Grant Davis generally provides feedback about hero mechanics, UI design and level design. Dan Huynhvo generally provides feedback about UI design, AI, and level design.

Going forward this is the general principle we will keep to ensure knowledge through the team is shared and our vision is consistent. Testing procedure shares this responsibility to all members for all parts of the project, however focus is given to the systems developed by each member. Having designed specific systems it is assumed that members would have a better understanding of what could go wrong and what the desired outcome is. This is also reflected in our defect and debugging policy. Members who have designed the system are generally responsible for fixing any issues found in their testing or the testing of other members. The reason for this being that it is assumed they possess the greatest knowledge of the functionality of their systems and the potential reasons behind any such defects.

The project is deemed complete and testing successful when we achieve success on our weekly goals as defined on our trello board prior to our meeting with teaching staff. Overall the project will be deemed complete when we have reached all functional and nonfunctional requirements with a degree of polish that satisfies each member of the team.

To test acceptance criteria of the UI elements, we would need to create UI elements on the screen that can be tested for functionality using unit or acceptance testing. To test the enemy acceptance criteria, we can implement enemy and enemy AI, and have users test it to verify if enemy interactions are diverse enough. Card system and level generation both need to be implemented first, then revised with user feedback, such as whether the amount of cards is enough. Asset creation and audio work flow are both implementable and observable, so to test those, user testing and feedback is best. Bug fixing, error handling, level generation, and performance are best tested by unit testing, which can output error messages when issues arise. AI behavior and level design are observational, so with user testing and feedback, they can be adapted to become more complex or challenging.

6. Unit Testing

Case 1 - Robert Bothne: Test to determine whether or not all Poses which serve as room instantiators have been destroyed. If the count does not match then we are not deleting all poses.

```
private void testPoseDeletion()
{
    if (poses == null || poses.Length == 0 || poses[0]==null)
    {
        Debug.Log("All Pose SHOULD be nodes destroyed");
    } else if (poses != null || poses.Length != 0)
    {
        Debug.Log("Not all Pose nodes destroyed:" + poses[0]);
    }
    int expectedDestroyed = 16;
    if (expectedDestroyed == posesDestroyed)
    {
        Debug.Log("Expected poses deleted, success");
    }
    else
    {
        Debug.Log("Poses not deleted correctly, failure" + posesDestroyed);
    }
}
```

```
UnityEngine.Debug.Log (object)
[18:29:50] Pose to DestroyPose (13)
UnityEngine.Debug.Log (object)
[18:29:50] Pose to DestroyPose (14)
UnityEngine.Debug.Log (object)
[18:29:50] Pose to DestroyPose (15)
UnityEngine.Debug.Log (object)
[18:29:50] Pose to DestroyPose (16)
UnityEngine.Debug.Log (object)
[18:29:51] All Pose SHOULD be nodes destroyed
UnityEngine.Debug.Log (object)
[18:29:51] Expected poses deleted, success
UnityEngine.Debug.Log (object)
```

Case 2 - Abida Mim: Test to determine if health value going into the SetHealth function is being correctly set, correct testing for this would have the health_recognized value which was returned from the function match the argument, which would display the debug message.

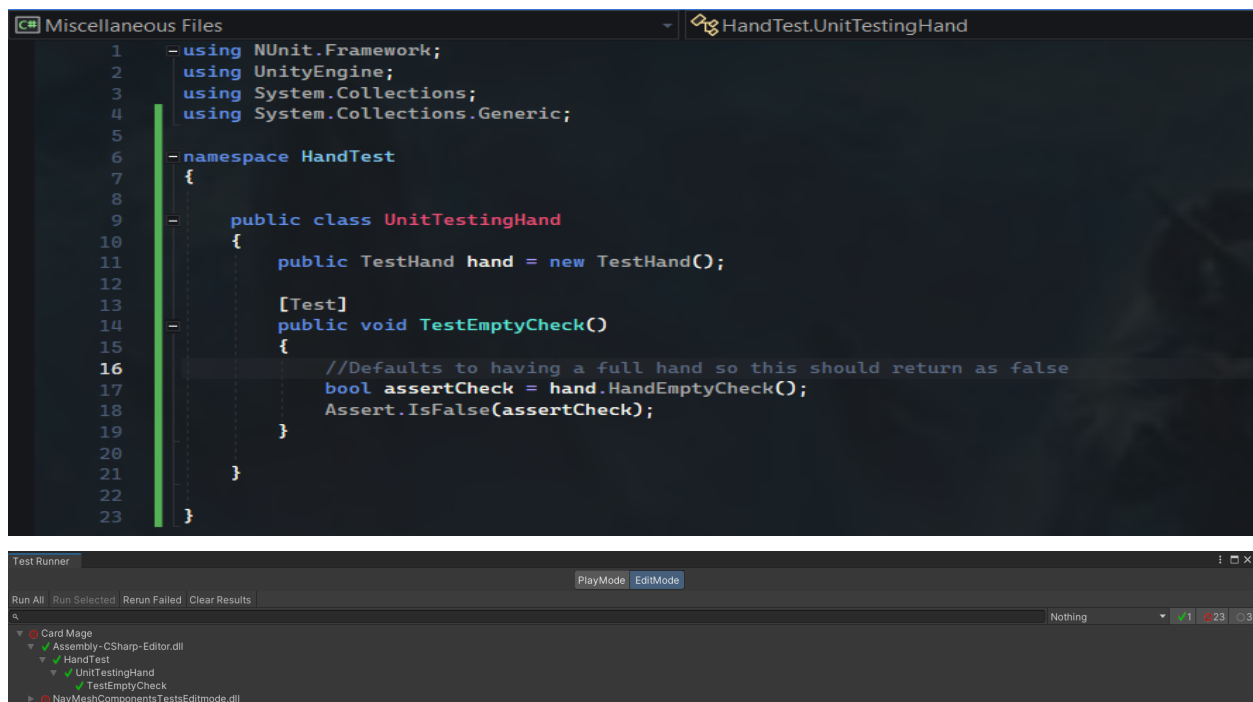
```
public void TestHealth()
{
    float health_recognized = HealthBar.SetHealth(80);

    if(health_recognized == 80)
    {
        Debug.Log("Health Bar is getting a value");
    }
    else
    {
        Debug.Log("Health value is not correct");
    }
}
```

[19:10:36] Health Bar is getting a value
UnityEngine.Debug:Log (object)

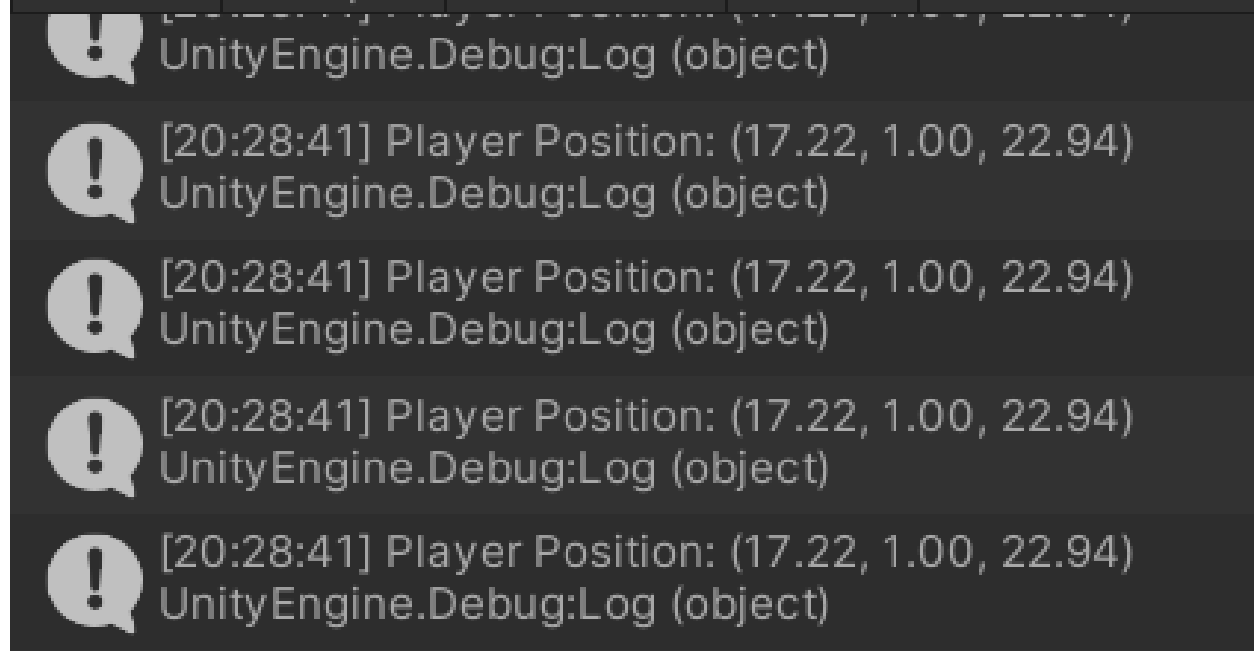
[19:10:35] Basic Played
UnityEngine.Debug:Log (object)

Case 3 - Dan Huynhvo: Test to determine if the player's hand is empty or not. If the player's hand has any objects in it at all then it should return as false. NavMesh was not being tested and can be ignored.



Case 4 - Grant Davis: To ensure the player character is correctly updating its position while alive and moving around the game world, perform a test to log the player's position in the debug log. If the player position is updating consistently and accurately in the debug log, it indicates that the player character is alive and moving as intended.

```
private void TestMovementTowardsPlayer()
{
    if (target != null && EM.NotDead)
    {
        float step = speed * Time.deltaTime;
        transform.position = Vector3.MoveTowards(transform.position, target.position, step);
        Debug.Log("Player Position: " + target.position);
    }
}
```



7. Contributions of team members

Robert Bothne: Project Updates and Changes. Unit Testing. Testing Strategy. User strategy and acceptance criteria - 3 hours

Grant Davis: Abstract, Testing Workflow, Unit Testing, Formatting/Editing - 3 hours

Dan Huynhvo: Unit Testing, User Stories, Project Updates and Changes, Editing - 3 hours

Abida Mim: User Stories and Acceptance Criteria, Unit Testing, Testing Strategy, Project Updates and Changes - 3 hours