Name: Mary Everett Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

My feature for the Tangent videogame is the math functionality and assignment for object creation and interaction.

When a level is generated, my job to make sure that the environment objects have a derivative or integral pulled from a pool of possible derivative and integral functions, and that an answer object that corresponds to the environment object is created with the correct match. I also need to make sure that the answer object is placed in the level before the environment object, so that the level is solvable.

Additionally, I’m responsible for making sure objects know if they are the correct object to interact with another object. If the objects are used correctly (that is, if the object comes into contact with the correct answer or function object), an event is triggered.

## Use case diagram with scenario \_\_14

### Use Case Diagrams

A close up of text on a white background

Description automatically generated

A close up of text on a white background

Description automatically generated

### Scenarios

**Scenario 1 (1st Use Case Diagram):**

**Name:** Assign Math to Object

**Summary:** The level generator system finds one out of a pool of correct math strings and assigns it to the object, while assigning a corresponding math answer object and ensuring both are at the correct position in the level.

**Actors:** Level Generator System

**Preconditions:** The basic object has already been created.

**Basic sequence:**

**Step 1:** The level generator specifies what type of math assignment is needed.

**Step 2:** Math tables are used to find a pool of appropriate math strings.

**Step 3:** A math string is chosen from the pool and assigned to the object.

**Step 4:** A math answer string is assigned to the corresponding answer object.

**Step 5:** The placement of the objects are checked to ensure that the answer object comes before its corresponding function object in the level.

**Exceptions:**

**Step 1:** A trigonometric math object needs assignment.

**Step 2:** A special trigonometric table is used to find an acceptable math string and assign it.

**Step 3:** A special trigonometric table is used to assign the corresponding answer object.

**Step 4:** Both object’s positions are checked.

**Post conditions:** The object has a math string corresponding to a math answer object, and is correctly positioned in the scene.

**Priority:** 1\*

**ID:** AM1

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

**Scenario 2 (2nd Use Case Diagram):**

**Name:** Interact with Math Object

**Summary:** The environment object checks to see if it has been presented with the correct answer object.

**Actors:** Player

**Preconditions:** Player has selected an input object.

**Basic sequence:**

**Step 1:** Check the input object against the math string.

**Step 2:** If the input is correct, trigger an event.

**Exceptions:**

**Step 1:** The wrong input is presented.

**Step 2:** No object event is triggered, but the input is rejected.

**Post conditions:** An event is triggered.

**Priority:** 2\*

**ID:** IM1

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

## Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_\_\_\_14

In the data flow diagrams below, I will be covering the Math Assignment Feature in entirety. I will describe the “Assign Environment Object” sub-process with a decision tree.

### Data Flow Diagrams

A close up of a map

Description automatically generated

### Process Descriptions

The process description for Process 4.2.1 (Assign Environment Object) is displayed below in a decision tree.

A close up of a map

Description automatically generated

## Acceptance Tests \_\_\_\_\_\_\_\_9

This feature has both pre-determined elements (math function type and difficulty level) and random elements (coefficients for the function and number of combinations for certain difficulty levels). The acceptance test will primarily test the limits of the random (but bounded) elements per math function type and difficulty level.

This feature will also need to be tested on whether or not it assigned the correct answer to the corresponding answer object, and whether or not it placed the answer object before the environment object in the level.

The Acceptance tests for these features are described below.

**Function Assigner:**

The feature will be automate to run 1000 times with each type of math function on each difficulty level (2 types \* 7 levels for derivative/integral, 1 type \* 6 levels for trig function, for a total of 20 tests). These will be send to an output file where the coefficients will be checked.

The output file should display the functions. The difficulty levels have different bounds for coefficients and math objects. No one coefficient should appear in the output more than 200 times (except for trigonometric functions), and each coefficient should appear at least once for every difficulty level. In the difficulty levels where the coefficients are restricted between -10 and 10 (excluding 0), each coefficient should appear at least 20 times. The exact allowable range of number of appearances for coefficients will vary based on the bounds imposed on the coefficients per difficulty level.

For combination levels, the combinations of functions will also be checked to ensure that no combination has more than 3 terms, and no combination term has the same power of x as another combination term. For trigonometric functions (which will not have coefficients in this game) combinations will be checked based on the difficulty level to ensure no trig function is used too often per difficulty level.

**Answer Assigner:**

The answer assigner feature will be checked at the same time as the function assigner. The function assigner will trigger answer assignment, and those answers will be put into a corresponding output file. These answers will be checked against the function/answer matrix to ensure the correct answers are being generated.

**Answer Placement:**

The position placement of the objects from the function and answer tests will be verified as well. A simple check that the x position of the answer object occurs below the x position of the environment object will be run on each output function.

**Example Output Test Check Analysis for Success/Failure:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Object | Difficulty Level | Coefficient(s) | Coefficient(s)/Trig Function Times Appeared | Function | Answer | Function Placement | Answer Placement | Success | Fail |
| Derivative | 2 | 5 | 180 | 5x | 5 | 121.00 | 76.00 | Yes | No |
| Derivative | 3 | 4, 7, 8, 9 | 140, 120, 80, 90 | 4x+7+8+9 | 4 | 356.21 | 312.58 | No | Yes |
| Integral | 1 | 2 | 150 | 4 | 4x + C | 51.94 | 1.2 | Yes | No |
| Trig Derivative | 3 | -- | 160, 170 | sinx + cosx | Cosx -sinx | 943.76 | 558.31 | Yes | No |
| Integral | 3 | 1 | 220 | X2 | (1/2)X3 | 45.00 | 82.11 | No | Yes |

## Timeline \_\_\_\_\_\_\_\_\_/10

### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (Hours) | Predecessor Task(s) |
| 1. Math Matrix Creation | 4 | - |
| 2. Math Matrix Data Structure Programming | 3 | 1 |
| 3. Math Superclass Design | 4 | 1,2 |
| 4. Math Subclasses Design | 6 | 3 |
| 5. Math Assignment Algorithm Design | 4 | 4 |
| 6. Match Checking Programming | 3 | 5 |
| 7. Placement Checking Programming | 6 | 5 |
| 8. Documentation | 3 | 6,7 |
| 9. Testing | 3 | 6,7 |
| 10. Installation | 3 | 8 |
| 11. Artwork | 6 | - |

### Pert diagram

A screenshot of a cell phone

Description automatically generated

### Gantt timeline

A screenshot of a cell phone

Description automatically generated