Name: Brandon Lunney

Total Marks __/50

1. Brief introduction __/3

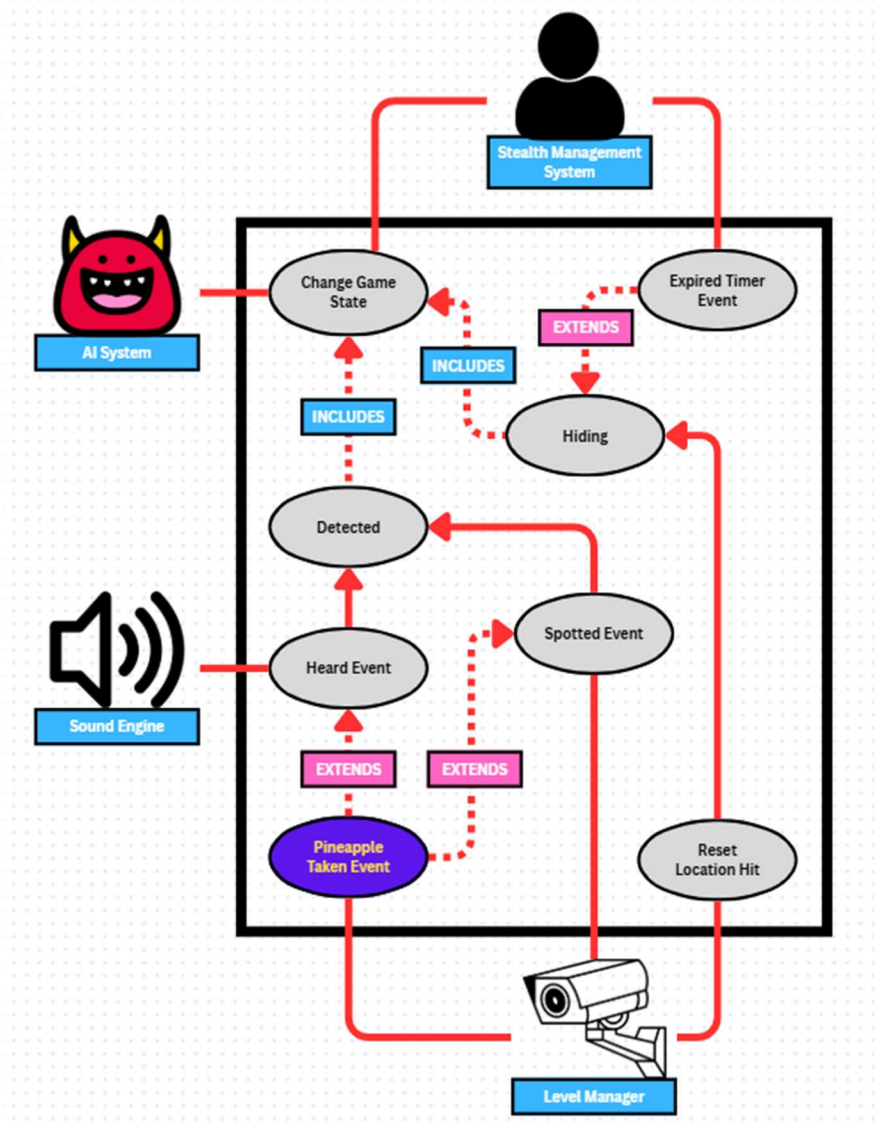
I will be responsible for implementing stealth game mechanics and interactions with other systems such as the level design, character, and enemies.

The system will rely on being within a specific “sight” range of a detection agent, making sound within “hearing” range of a detection unit, and either resetting state via checkpoint or elapsed time.

I will integrate the connections between systems within the level design for sound traps and the signal system to interact with detection agents within the play space. This will require partnership and coordination with other game systems being designed by other teammates.

2. Use case diagram with scenario __/14

Use Case Diagram



Scenario Concept

Scenario 1: **Change Game State**

Summary: The Game State is changed and updates the AI System

Actors:

- Stealth Engine
- AI System
- Level Manager
- Sound Engine

Preconditions:

- Detected or Hiding event triggered.

Basic sequence:

1. Determine current state from Sound Engine, Level Manager, State, and AI System Data.
2. Update State Data.
3. If Hiding, send state change to AI System to switch all enemies to Oblivious.
4. If Detected, start or reset timer.
5. If Detected, Determine alerted enemies.
6. Send effected enemy states to AI System.

Exceptions:

2. If pineapple is taken, change all enemy states to alert.

2. If timer expires, send hiding command.

Post conditions: Mode is correct and enemies are either alert or oblivious based on system inputs.

Priority: 1 [Must Have]

ID: GM1

Scenario 2: **Detected**

Summary: Determine what kind of detection has happened and send that information to Change Game State

Actors:

- Level Manager
- Sound Engine

Preconditions:

- Heard Event, Spotted Event, or Pineapple Taken Event occurred.

Basic sequence:

1. Determine what event took place (heard, spotted, pineapple taken).
2. Send relevant data to Change Game State.

Exceptions:

1. If pineapple is taken, send only this information to Change Game State

Post conditions: Parsed relevant information is sent to Change Game State

Priority: 1 [Must Have]

ID: GM2

Scenario 3: Hiding

Summary: Set and send data relevant to changing to hidden state.

Actors:

- Stealth Engine
- Level Manager

Preconditions:

- Reset Location hit or timer expired.

Basic sequence:

1. Determine if time expired or reset location found.
2. Set relevant parsed data and send it to Change Game State

Exceptions:

1. If timer expires, send timer expired data straight to Change Game State

Post conditions: Hidden state is sent to Change Game State

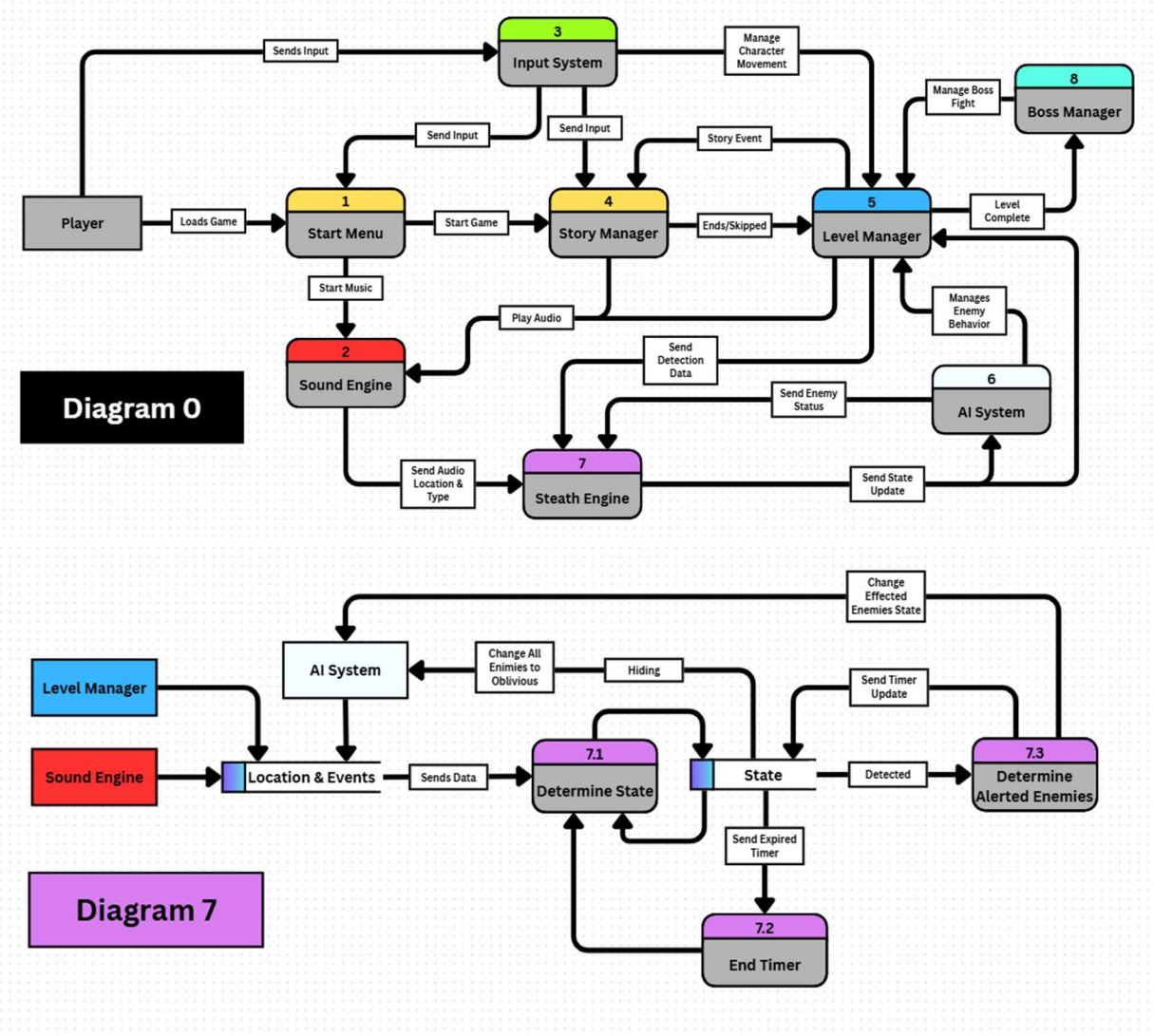
Priority: 1 [Must Have]

ID: GM3

3. Data Flow diagram(s) from Level 0 to process description for your feature

__/14

Data Flow Diagrams



Process Descriptions

Determine State*:

Pull Location & Event data.

IF timer expired **OR** event data is "hiding" **THEN**

Send hiding to state data.

Send signal to AI SYSTEM to change all enemies to oblivious.

END IF

ELSE IF event data is "detected" **THEN**

Send detected state to data.

Transition state to Determine Altered Enemies.

Pass Location & Event data.

END IF

Determine Altered Enemies*:

SET timer to default time.

FOR enemies **IN** location data

IF location is **EQUAL** to alerted range, **THEN**

Place enemy in alerted enemy list.

END IF

Send altered enemy list to AI System.

Determine End Timer*:

Pull timer data from State.

IF timer expired **THEN**

Send timer expired to Determine State.

Transition state to Determine State.

END IF

ELSE IF timer data is reset **THEN**

Transition state to Determine State.

END IF

4. Acceptance Tests ___/9

Stealth Management System

This system has three states, two data locations, an internal timer, and three input streams. As such the tests will use these locations as attack vectors for tests to be executed 1,000 times to test validity. If a failure state is found, it will be logged and the test process will terminate for that run.

Testing Structural Validity

These are the tests that will be run:

- Ensure states do not lead to dead ends.
- Validate that the timer behaves and triggers correctly.
- Send junk data to the date locations to ensure proper data sanitization.
- Ensure bad data does not break the state flow if unexpected data is pulled.

Example statistics for generated maps

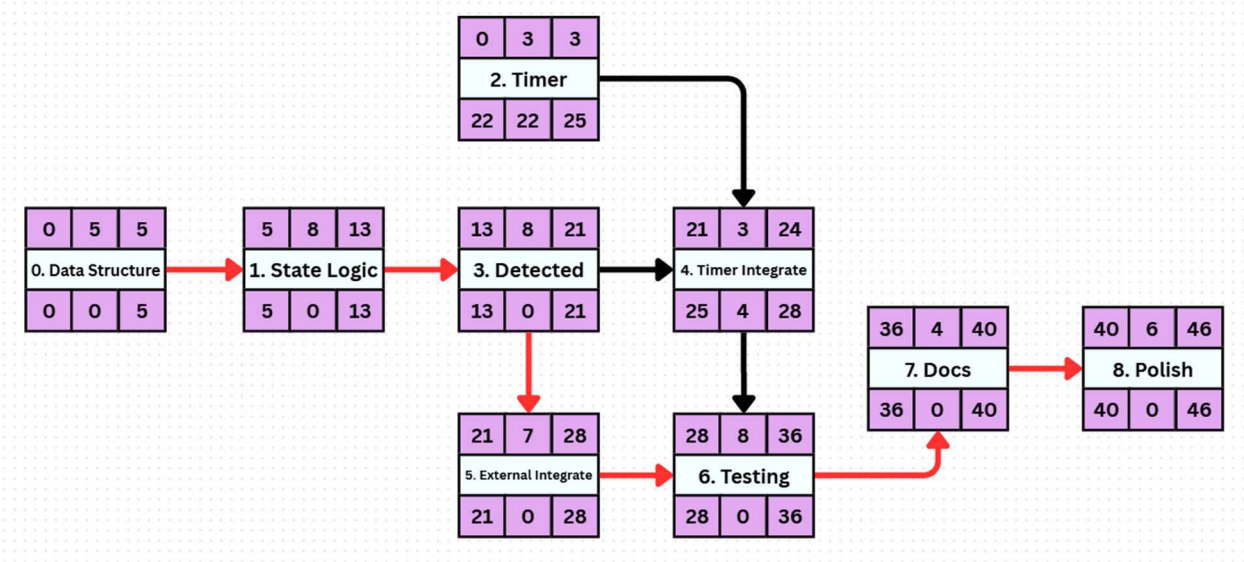
Statistic	Value	Pass Tests?	Notes
Timer Failures	FAST COMMAND	T	Sent repeated commands for reset, end, and start in random sequences. Behavior as expected.
Start event connections	T	T	Tested starting at different states to see if a failure could be reached
Send random junk as input to data locations	RANDOM INPUT	T	Flood the data locations with junk input. No failures detected.
Pull junk data from data	JUNK DATA	F	Had states pull data from poisoned data set to ensure second catch for data corruption, failure detected in 7.3

5. Timeline __/10

Work items

Task	Duration (HRS)	Predecessor Task(s)
0. Receiving Data Structure	5	-
1. Determine State Logic	8	0
2. Background Timer	3	-
3. Determine Alerted Enemies Process	8	1
4. Timer Integration	3	2, 3
5. External Input System Integration	7	3
6. Testing	8	0, 1, 2, 3
7. Documentation	4	0, 1, 2, 3
8. Polish	6	7

Pert Diagram



Gantt Timeline