TECHNICAL DESIGN DOCUMENT:

Instructor: Prasad Naveen

Game Title: Cheese Escape

Author: Kadeem Cherman

Course: VGP225 – AI Game Programming

Table Of Contents:

Game Overview:

Cheese Escape A top-down stealth and puzzle-based game where the player controls a mouse trying to retrieve cheese at the far end of a map while avoiding detection and capture by patrolling AI-controlled cats.

The player navigates around obstacles, uses physics-based traps (like bounce pads and force fields), and outsmarts AI enemies with there clever AI behavior to complete each level.

This project is designed to demonstrate both AI and physics-based gameplay elements in a beginner-friendly scope with a win/loss outcome.

CONCEPTS USED FROM CLASS:

AI Concepts:

1. Finite State Machine (FSM):
   Each enemy cat uses an FSM to switch between *Patrolling*, *Chasing*, and *returning to Patrol* states depending on player visibility and distance.

2. A Pathfinding (Unity NavMesh):
   Cats use the built-in Unity NavMeshAgent to dynamically navigate around obstacles and pursue the player using A* pathfinding-like behavior.

PHYSICS CONCEPTS:

1) Kinematics:
   The player's movement is handled using Rigidbody-based velocity to simulate acceleration and direction changes.

2) Collision/ Gravity Traps:
   The game includes bounce pads and force zones that push or pull the player

or enemies when they interact with the trigger volumes, showcasing practical use of force vectors and gravity.

Player Goals & Conditions:

- **Objective:** Reach the cheese located at the far end of the level.
- **Win Condition:** Player touches the cheese object without being caught.
- **Lose Condition:** Player is caught by any cat (enemy collision with player).

Project Setup/ Folder Structure:

Assets – Containing project structure folders:
Scripts – Containing game logic / Mechanics
Prefabs – Any game objects turned into prefabs
Materials – Colors/ Textures used for game elements
Scenes – Level Scenes
UI – Elements of the UI
NavMesh – Level design

Task Breakdown & Time Estimates:

Tasks #:	Task Descriptions:	Estimated Time:
1	Setup Unity Project Initial & Folders Structure, along with GitHub repo.	15 Mins
2	Create a player object with Rigidbody movement	45 Mins
3	Design enemies' prefab with FSM & NavMesh	2 Hours
4	Create patrol points & detection logic (RayCasting)	1.5 Hours

5	Create cheese goal object & win condition logic.	30 Mins
6	Add gravity-based traps (bounce pads, push/pull zones)	1.5 Hours
7	Design level layout with walls, paths, obstacles.	1 Hour
8	Build basic UI for win/lose affects.	1 Hour
9	Play Testing & debugging / code refactoring.	1.5 Hour – 4 Hours estimates depending on the debugging process.
10	Polishing the game & preparing for submission.	1 Hour – 2 Hours
11	Final Checks & playable copy along with build copy.	30 Mins

Total estimated time of development: [15 Hours]

Proto-Typing Feature: Enemy FSM + Pathfinding

For the first prototype, I implemented a **basic enemy AI FSM system** combined with Unity's NavMesh pathfinding.

- Cats patrol using an array of waypoints.
- If the player enters their sight range and is visible via raycast, the enemy switches to *Chase* state and updates destination via NavMeshAgent.
- If the player escapes their range, the cat returns to patrolling.
- This demonstrates both FSM logic and Unity's implementation of AI style navigation.
- 1 Player Capsule.
- 1 Cheese goal object.

- 1 Enemy cube patrolling 3 waypoints.
- NavMesh backed on a floor plane.

Next Iteration Steps:

Multiple enemy variations with different detection and speeds

Bounce pads and force zones with rigidbody-based propulsion

Maybe design more levels with increases difficulty.

Polish up more the material assets.