

Technical Design Document

Contents

Game Details.....	2
Team Members.....	2
Game Concept.....	2
Technical Risks.....	2
Technical Risk 1 – AI.....	2
Technical Risk 2 – Perforce.....	3
Features/Mechanics/Tasks.....	3
Deliverables.....	5
System Requirements.....	5
Target Device 1 - PC.....	5
Target Device 2 - Android.....	5
Third Party Tools.....	6
File Formats.....	6
Coding Conventions.....	6
Source Control.....	7
Game Flow.....	7
Game Objects and Scripts.....	8
Gameplay Systems.....	11
Gameplay System 1 – Flocking and Herding AI.....	11
Gameplay System 2 – Hazards and Threats.....	11
Gameplay System 3 – Player Control System.....	11
Input Method(s).....	12
User Interface.....	12

Game Details

- **Game Name:** *Paddock Panic*
- **Team Name:** *Flock Forge*

Team Members

Name	Job Title	Responsibilities/Roles
Alisla Foley (Alis)	Programmer	AI and Player controls
Melissa Scott	Designer / Programmer	Player Hazards and Incentives
Chelsea Siemers	Designer / Programmer	UI and Scoring

Game Concept

Paddock Panic is a hyper casual mobile game in which the player attempts to herd a flock of sheep through several hurdles and threats in the shortest amount of time as possible without losing them. The player achieves this through physically herding the sheep and barking at threats to bring them to their paddock safely. There are multiple pre-designed stages that the player must fully clear in order to win the overall game.

Technical Risks

Technical Risk 1 – AI

What's the risk about: AI is difficult to implement and can easily be broken and filled with bugs.

How will risk be mitigated: AI is an integral part of the game and everything is built around it, so it will take up most of the programming time. We will begin working on the AI immediately and focusing on perfecting it before moving on to other tasks.

Technical Risk 2 – Perforce

What's the risk about: Some programmers have not used Perforce as version control before.

How will risk be mitigated: Spending time getting used to the software and doing a rough test project before developing the main game to help with committing and using version control will help prevent any issues.

Features/Mechanics/Tasks

A list of exactly what systems exist in your game and who is responsible, including scheduled dates for completion.

Feature/Mechanic	Who's responsible	Description	Scheduled Date
Player walking (High priority)	Alis	The player drags to walk; will be completed using splines that the player follows once drawn using the mouse.	28/10/24
Player barking (High priority)	Alis	The player presses the left click button to bark, this is hard coded into the player character blueprint.	25/11/24
Sheep flocking AI (High priority)	Alis	The sheep have three different states, herded or individual. This will be coded using a flocking algorithm implemented through Unreal Engine blueprints, blackboard and behaviour tree. This is explained further in depth in Gameplay Systems below..	25/11/24
Wolf AI (High priority)	Alis	Wolves will appear randomly to stalk for a certain amount of time before kidnapping sheep. They are scared off by the player barking at them. Coded using blueprints, blackboard and behaviour trees.	25/11/24
Bee swarm AI (Has been cancelled)	Alis	A swarm will randomly spawn and chase sheep (possibly prioritizing ones in herds), making them panic and leave their herd (run away in a random direction, could possibly run into another hazard). Coded using blueprints, blackboard and behaviour trees.	12/11/24
Cliff (High priority)	Chelsea	Sheep will be permanently lost (killed) if they run off a cliff. A collision box near the cliff will immediately make the sheep walk off and die.	12/11/24
Broken fences (Low priority)	Chelsea	Sheep disappear permanently if they go through a broken fence. A collision box will make any nearby sheep walk through the fence and disappear off-screen.	12/11/24
Tractor / farm equipment (Medium priority)	Chelsea	Equipment moves around the map and destroys sheep permanently if they come into contact. Another collision box that immediately kills any sheep it comes into contact with (plays animation of being run over).	12/11/24
Falling tree branch (High priority)	Chelsea	Random chance for a tree branch to fall and separate the herd. A collision box on the falling branch will make a physical barrier between the sheep and separate them.	12/11/24

Streams <i>(Medium priority)</i>	Chelsea	Slows down sheep that are moving in the stream. A collision box in the stream will directly reduce the sheep's movement speed.	12/11/24
Falling rocks <i>(High priority)</i>	Chelsea	If a sheep is caught under a falling rock they are permanently killed. Works similar to the falling tree branch, where a collision box will kill any sheep it comes into contact with while falling.	12/11/24
Weather <i>(Medium priority)</i>	Chelsea	Fog that reduces the player's vision during bad weather conditions.	12/11/24
Score boost <i>(High priority)</i>	Chelsea	An in-game pickup that adds one star to the player's score; the total number of stars are shown at the end of the level. A collision box that once the dog touches will increase the player's score by one star.	12/11/24
Dog speed up <i>(Medium priority)</i>	Chelsea	Dog has +%50 more speed. ~10 second duration. A collision box that increases the dog's speed for 10 seconds.	12/11/24
Sheep magnet <i>(Low priority)</i>	Chelsea	An in-game magnet that makes sheep follow the dog when within a certain range. ~10-15 second duration. A collision box that applies a magnet effect to the dog for ~10-15 seconds (random).	12/11/24
Sheep armor <i>(Medium priority)</i>	Chelsea	The first sheep that runs into this pickup has immunity to the first hazard that they come into contact with. Lasts until they lose the buff to a hazard. A collision box that applies armor to the first sheep it touches.	12/11/24
Brief hazard removal <i>(Medium priority)</i>	Chelsea	Hazards become harmless for a short amount of time. ~10-15 seconds. A collision box that applies the hazard removal effect once the dog touches it.	12/11/24
Megaphone <i>(High priority)</i>	Chelsea	Dog's bark range increased by +%50. ~10 second duration. A collision box that applies the megaphone effect once the dog comes into contact with it.	12/11/24
Store <i>(High priority)</i>	Melissa	In the main menu where players can buy cosmetics for the dog and sheep. Will be another section implemented using Unreal Engine's built in UMG system.	26/11/24
Time limit <i>(High priority)</i>	Melissa	Game will visually change as the player runs out of time, from morning to sunset to night. Must return the sheep to the paddock before night or it is game over. Time limit between 30 seconds - 1 minute, will be adjusted depending on level. On screen clock will display how long the player has before nightfall. Will be implemented using Unreal Engine's blueprints and the UMG system for the UI clock.	24/11/24
Scoring <i>(High priority)</i>	Melissa	There are 3 stars and a bonus star that make up the overall player's score. Players need 3 stars to receive rewards, but can still proceed to the next level with 1 or 2 stars. There are penalties that reduce the score, such as crops that can be trampled. Unreal Engine's blueprints will be used to create this feature.	24/11/24

Deliverables

What will you deliver at the end of production?

Deliverable	Who's Responsible	Who's the Owner
Executable for PC	Alis	Flock Forge
(Has been cancelled) Executable for Android	Alis	Flock Forge

System Requirements

What devices is your game targeting? What's the recommended hardware? Portrait or Landscape mode on mobile?

Target Device 1 - PC

Recommended Hardware: *Mid-range gaming PC with at least 8GB RAM, Intel i5 Processor, and GTX 1050 or higher.*

Platform Specific Requirements: *Mouse and keyboard control required for line drawing and barking actions. The game will be played in portrait mode with a fixed resolution size of 640 x 1136.*

Target Device 2 - Android

Recommended Hardware: *Android devices with 4GB RAM and above*

Platform Specific Requirements: *Portrait mode, responsive to touch controls for movement and interaction. Also with a recommended resolution of 640 x 1136, but can be slightly stretched to match the resolution of other mobile device screens.*

Third Party Tools

Game Engine: Unreal Engine 5.4.4

Version Control: Perforce, GitHub

3D Models/Assets: Maya, ZBrush, Substance Painter

AI Assets/Tools: TBA (currently none)

File Formats

Models: .fbx, .obj

Textures: .png, .jpg

Sounds: .wav, .mp3

Level Data: Custom level format for Unreal Engine (.umap)

Coding Conventions

- **Follow Unreal Engine coding standards:**

Correct prefixing for classes (U for UObject and A for AActor), CamelCase for function names and PascalCase for class names.

- **Adopt best practices for AI and character behavior implementation:**

Reduce the amount of Tick functions for AI characters as they are called every frame. With a large amount of characters, this will greatly impact the games performance. Behaviours will be triggered based on Unreal's Blackboard key values instead of checking every frame for changes.

- **All team members must conform to agreed guidelines for naming conventions, comments, and code structure:**

- PascalCase for class and structure names with prefixing to indicate the class type (e.g: AEnemyWolf [A for AActor]).
- PascalCase for functions with names that accurately describe what the function does (e.g: ChaseClosestSheep();).
- Use BP_ as a prefix for Unreal Engine blueprint classes to help differentiate them from C++ classes.
- In C++ files, comment at the top of each file to describe its purpose, who created it and when it was created (e.g: 'EnemyWolf.h', 'This class represents the wolf enemy in game, it determines its movement, combat and interaction with the sheep and dog.', 'Created by: Bob', 'Date: 22/10/24').
- Comment above functions to describe their purpose, parameters and return values (e.g: 'ChaseClosestSheep();', 'Parameters: Sheep - the sheep to be chased', 'Return: Boolean - has a sheep been successfully found and a chase has begun?').
- In Blueprints, use comment boxes to describe nodes and their purpose (e.g: 'Movement - these nodes control the player's movement').

Source Control

Source Control Repository: Github and Perforce

Source Control Client Tools: Perforce P4V, connected to Unreal Engine's Revision Control

Source Control Remote Repo URL: <https://github.com/alisthela2/PaddockPanic>

Ignore/Config file: <https://github.com/alisthela2/PaddockPanic/blob/master/.gitignore>, for Perforce only selected folders/files are uploaded (config, content, source and .uproject).

Commit Rules:

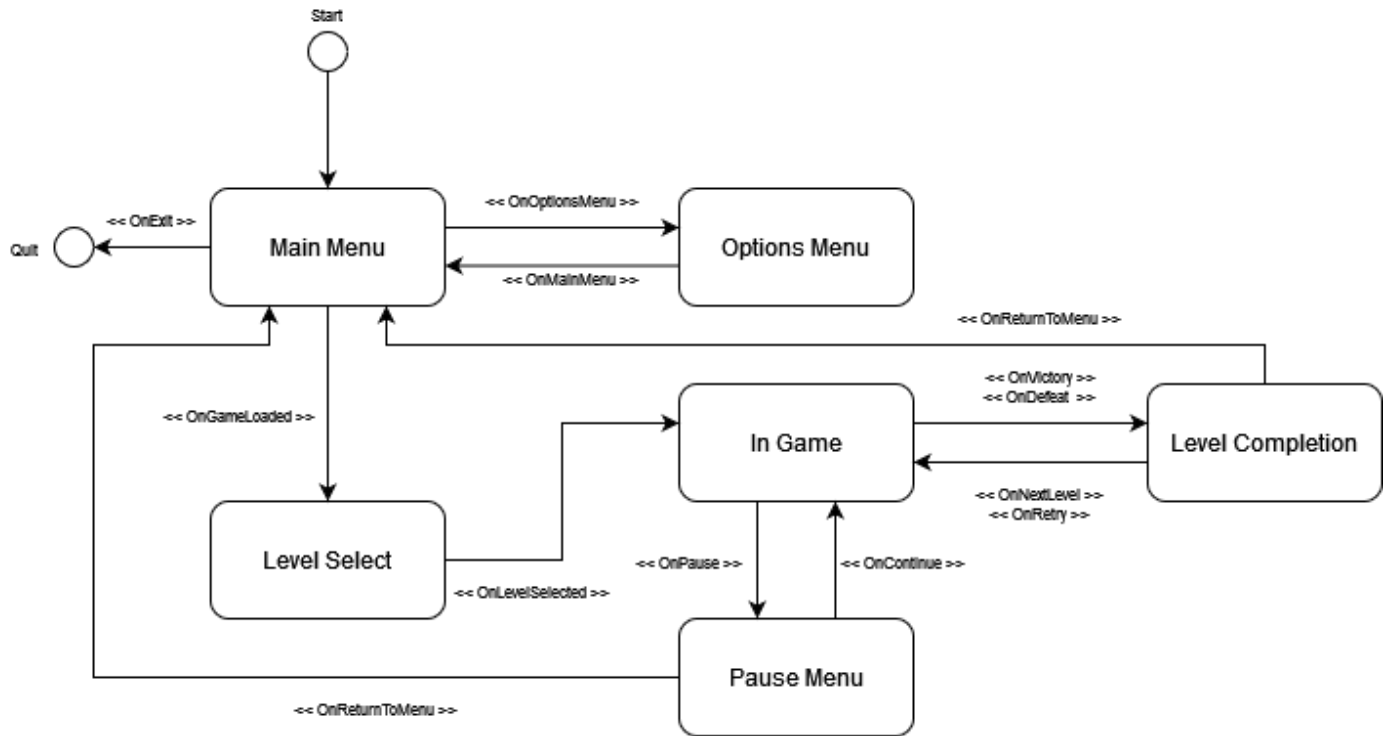
- Regular commits, with each change tagged with clear messages. Use separate branches for major features. Use a standard format like fix: description of fix or feature: description of new feature.
- Sign your name at the bottom of the commit and mention what should be worked on next in a 'TODO:' section.
- Any merge conflicts will be merged with both programmers going through and choosing what to keep.
- Further in development during the Beta, do not commit any changes that are works in progress and may break the game.
- Any issues with conflicts or files going missing/deleted and not being able to be returned, contact teachers or AIE online administrators.

Game Flow

List the scenes in the game, and a short description of what the scene is responsible for.

Scene	Who's responsible	What it does
Main Menu	Melissa	First screen in the game, features a new game, load game, settings and exit button.
Level Select	Melissa	After the player presses a new game or load game, this scene will be shown. Determines what level the player will be loaded into.
Options Menu	Melissa	Includes options to change the volume and brightness.
Game Level	Melissa	Loads the previously selected level where gameplay takes place. Players can access the pause menu, change their options and return to the main menu.
Level Complete	Melissa	Shown on level completion, provides the player with their score and an option to retry or continue (loads the next level).

Diagram on next page =>



Game Objects and Scripts

Game Object / Script	Who's responsible	Description
Dog	Alis	Will be implemented using Unreal Engine's blueprints. The player drags using the left mouse button and a spline will be drawn. The dog will subsequently follow that path. If there is an object in the way of the dog following that path then it will not move. This promotes the player being intuitive and instead drawing paths around the hazards.
Sheep	Alis	The sheep follow three main states, being herded, grouped or individuals. When the dog is nearby they move in the opposite direction which can be coded using blueprints. The grouped behaviour will be implemented using a flocking algorithm, which will use an AI behaviour tree and the blackboard for its keys. When a key value changes, such as 'IsInHerd' or 'IsDogNearby', the behaviour tree will run certain tasks like 'RunAway' or 'HerdRoam'.
Wolf	Alis	The wolf has a stalking state, an actively hunting state and a running away state. After a certain period of time passes, the wolf switches from stalking to hunting and attempts to kidnap the nearest sheep. However, if the player barks at the wolf in time it will run away. This will be implemented using a blackboard and behaviour tree that switches between the three states depending on boolean key values such as 'DidDogBark' and 'IsHoldingSheep'.

Pickups	Chelsea	Pickups will be objects in the world that have a collision box that apply their effect when they come into contact with the player or, for the hazard protection, a sheep. These effects will most likely be coded using Unreal Engine's blueprints.
Obstacles/Threats/Hazards	Chelsea	Sheep will avoid obstacles with their pathfinding. Threats and hazards will mostly be collision boxes that immediately kill the sheep when they come into contact. Each has their own gimmick, such as the tractor moving around and the falling branch only killing when falling on the sheep. The only non-lethal hazard is the stream, which will be a collision box that reduces the sheeps speed.

Class List (Player and AI)

Class	Connected Classes	Description
BP_Dog	BP_Path	Controls the player's movement (drag to move, click to move), barking, hazards that affect the dog, related animations and sounds as well as cosmetics. Also includes a camera on a spring arm to control the isometric angle and zoom range.
BP_Path	BP_Dog	Reads the path Vector3 mouse position values passed in from BP_Dog and converts them to spline points. A particle system is played along the spline to simulate a path being drawn.
BP_Sheep	BP_SheepAIController, BP_SheepBehavior	Controls sheep flocking behaviour, hazards, incentives, reaction to being kidnapped by a wolf, death, related animations and sounds as well as cosmetics. It also contains HUD elements such as the behaviour indicator icons above their head and off-screen indicators.
BP_SheepAIController	BP_Sheep	Has functions that allow the sheep to roam around, run away from the dog and bounce off of walls. These are called from BP_SheepBehaviour, though BP_SheepAIController has no reference to it.
BP_SheepBehaviour	BP_Sheep, BP_SheepAIController, BP_SheepBlackboard	The behaviour tree that determines which tasks the sheep will currently run. Tasks run functions from BP_SheepAIController if they are complex, but otherwise coding is done in the task itself. Booleans are taken from BP_Sheep and stored in the BP_SheepBlackboard.

BP_SheepBlackboard	BP_SheepBehaviour	Stores values from BP_Sheep such as the booleans 'IsDogNearby', 'InFlock' and 'CanMove' as well as game objects like 'SheepObject' and 'KidnappingWolf'.
BP_Wolf	BP_WolfAIController, BP_WolfBehaviour, BP_WolfSpawnPoint	Controls the wolf's stalking state, detects nearby sheep victims and finds the closest one, finds the closest spawn point (also exit point), death and a successful kidnap function that calls the kidnapped sheep and wolf's death function. There is also an off-screen HUD element that shows the wolf's location.
BP_WolfAIController	BP_Wolf	The AI controller includes the ability to have the wolf stalk sheep and chase sheep.
BP_WolfBehaviour	BP_Wolf, BP_WolfAIController, BP_WolfBlackboard	The behaviour tree that determines which tasks the wolf will currently run. Tasks run functions from BP_WolfAIController if they are complex, but otherwise coding is done in the task itself. Booleans are taken from BP_Wolf and stored in the BP_WolfBlackboard.
BP_WolfBlackboard	BP_WolfBehaviour	Contains Booleans, Objects and Vector3 values that are retrieved from BP_Wolf and BP_Sheep. These are passed to BP_WolfBehaviour to control which tasks should be run.
BP_WolfSpawnPoint	BP_Wolf	Although not directly connected, BP_WolfSpawnPoint spawns an instance of BP_Wolf in the level. When the wolf kidnaps a sheep it runs towards this point and exits.

Gameplay Systems

Gameplay System 1 – Flocking and Herding AI

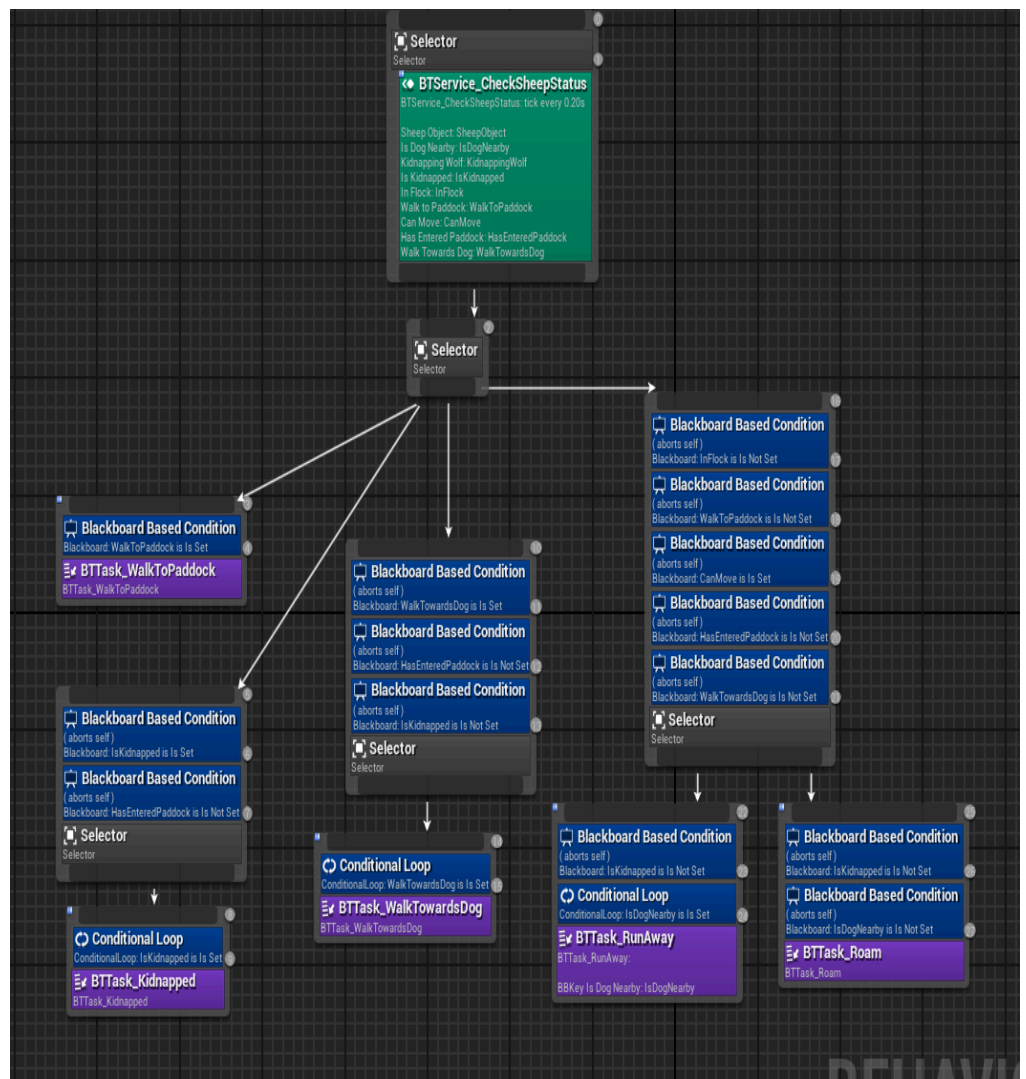
Who's Responsible: Alis

Description: Sheep Behaviors: Sheep can behave in three states (Herded, Group, or Individual) based on proximity to the dog and other sheep.

- Herded sheep move away from the dog.
- Groups of sheep move in unison, while individual sheep have random grazing and moving behaviors.
- Threats like wolves can force sheep to be lost.

Sheep flocking was added by following an Unreal Engine 4 document created by Peter L. Newton and Jie Feng called 'Unreal Engine 4 AI Programming Essentials', specifically 'Chapter 7: More Advanced Movement'. There are three main values that determine the specifics of the sheep flocking behaviour: cohesion, alignment and separation. These are calculated based on their respective default values multiplied by equations for each value. Cohesion is what keeps the sheep together in a group, alignment is based on nearby sheep and aligns their forward facing direction and separation keeps sheep at a certain distance from each other. Flocking behaviour activates when 3 or more sheep are within 1000 units of each other, setting a bool called 'InFlock' to true. This bool prevents two tasks from running inside the sheep's behaviour tree, 'Run Away' and 'Roam' as these behaviours are specifically for individual sheeps. Instead, the flocking algorithm that runs off of Event Tick decides the sheeps roaming movement direction and how they run away from the dog.

When certain blackboard booleans are set to true or false, different behaviour tree tasks are activated that affect both individual and flocking sheep. When a wolf has kidnapped a sheep they are put into a kidnapped state (conditional loop on the task 'Kidnapped' based on the bool 'IsKidnapped'), which causes them to be attached to the wolf's mouth and rotated upside down. When the dog barks at a nearby sheep, a boolean called 'WalkTowardsDog' is set to true, which makes the sheep follow the dog and stop when within 500 units for 1 second, then return to its usual behaviour of either flocking, roaming or running away if the dog is still close enough.



Gameplay System 2 – Hazards and Threats

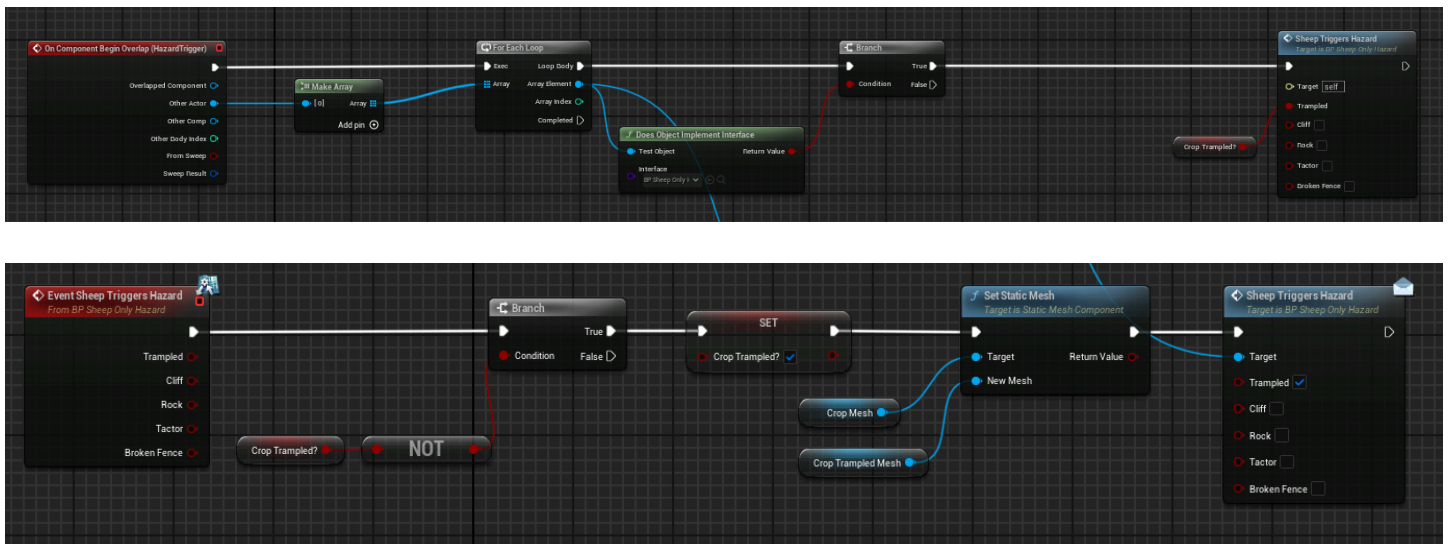
Who's Responsible: Chelsea and Alis

Description:

- Cliffs and Wolves, etc: Hazards reduce the player's flock, with sheep falling off cliffs or being lost to predators.
- Wolf AI: Wolves will appear on the map, and the player must bark quickly to scare them off.

All hazards, except the wolf, are implemented using two blueprint interfaces called 'BP_Hazard' and 'BP_SheepOnlyHazard'. Hazards function by sending a message to the object that activated it, usually through collision. The message activates a bool, which in 'BP_Hazard' there is only one that slows the dog, wolf and sheep down when walking through a stream. In 'BP_SheepOnlyHazard' there are bools for the sheep falling off the cliff, walking through broken fences, getting run over by a tractor and falling rocks. When these bool values are changed through a message, the sheep run their respective response, such as playing an animation, smoke particle effect and then being removed from the game.

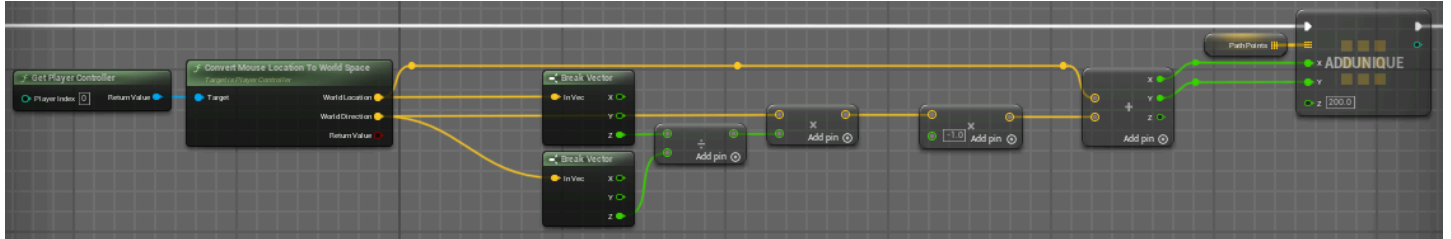
Example for crops being trampled shown below, where a message is then sent to the BP_Sheep object that triggered the hazard:



Gameplay System 3 – Player Control System

Who's Responsible: Alis

Description: Players click and drag to draw a line that the dog follows or they can press left click to move towards the click position. When the player is holding left click, the mouse position is recorded every tick and saved in an array of Vector3 values. On the release of left click the dog begins to follow the mouse position values which are passed to another blueprint that draws the path. The path is drawn by adding points to a spline based on the mouse position values and enabling a Niagara Particle System that follows the spline.



The movement system can be found in the BP_Dog blueprint and the drawn path can be found in BP_Path.

The dog can bark by pressing right click, scaring away nearby threats like wolves and enticing close sheep to walk towards the player for roughly 1 second. The wolf completely stops any behaviour tree task it is currently doing and immediately begins the 'Run Away' task due to the bark setting a bool in the wolf's blackboard called 'ShouldRun' to true.

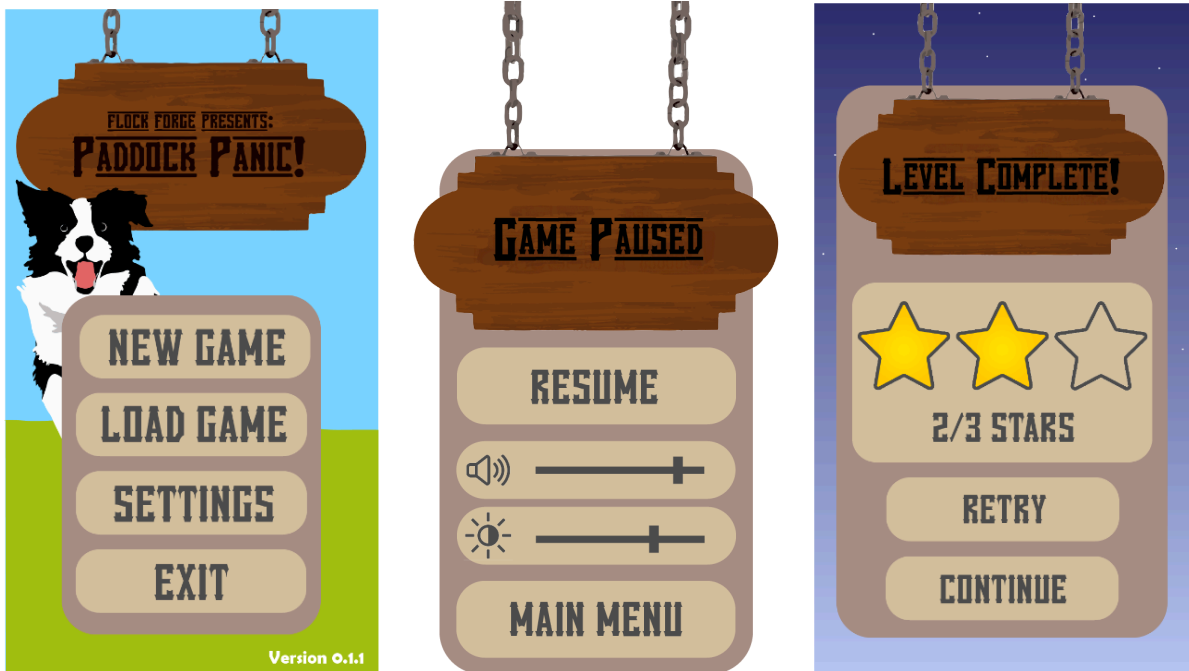
Input Method(s)

Describe the Input method for each target platform (e.g PC / VR / Console).

Target Platform	Input System	Who is responsible
Android (has been cancelled)	Touchscreen	Alis
PC	Mouse	Alis

User Interface

Add user-interface design mockup. Display any differences between PC and Mobile interfaces.



There will be no differences between PC and Mobile interfaces.