School of Computer Science and Informatics

MSc Computing

Dissertation

CMT400

Exploring the potential of a serious gaming approach to address antibiotic use in livestock production

Author

Omid Mansour

1953084

Supervisor

Nervo Verdezoto Dias

October 2nd 2020

Acknowledgements

I would like to thank my supervisor Dr Nervo Verdezoto for his excellent supervision and guidance. Thanks to Matt and the rest of the team who have also assisted the project with their expertise on the topic.

Abstract

The topic of human antimicrobial resistance (AMR) benefits from innovative ways to express the consequences of human practice. One such way includes the use of serious games to spread awareness. Applications of serious games in clinical use have extended to teaching kids the importance of hygiene through educational games as well as professional training games to train medical students. However, humans are not the only ones affected by the consequences of these practices. There is a lack of awareness on practices regarding treating livestock with antibiotics. Moreover, there is a distinct lack of serious games on the topic of AMR in agriculture. To explore the potential benefits of applying innovative approaches on AMR in, an initial prototype for a game featuring animal health management had been developed using the Unity 3D engine. Farmers and vets have been approached with this prototype of a serious game to gauge their interest. To understand how this serious game could be purposed, the prototype had been designed with the User Centered Design methodology. *Where's Woolly*, the game, had been tested and evaluated and found that the design and core gameplay was enjoyable. Our participants had realized the purpose could be education and training. There had also been beneficial feedback about the future direction.

Contents

Figures

Tables

1. Introduction

Serious games - games which combine play with industrial utility without a sole focus on entertainment (1) - have recently been applied for education and communication around the issue of antimicrobial resistance (AMR). Due to the impacts of AMR, being invisible and gradual, AMR needs innovative ways to describe the long-term issues regarding consequences of practice through engagement, hence serious games might be appropriate. Games about antibiotic use and resistance have seen success as educational training tools, (e.g., *Prognosis: Your Diagnosis*) as well as entertainment, (e.g., *Superbugs*). However, there is practically no gamified representation of the complications of farm animal health management, specifically antibiotic resistance. This is despite the fact that globally, over 50% of a country's antibiotics are used on farm animals (2) and pressure to reduce antibiotic use on livestock production is growing. This project explores the broad potential of gamification of agricultural AMR and how this approach might be used to engage with the issue of antibiotic use in livestock production.

The scope of the project is to build a prototype game and, in the process, explore potential uses for such a game about antibiotic use in livestock production. The aim is to explore the feasibility of implementing, antibiotic practices as a serious game in a way to which farmers can relate. The project investigates the potential uses of the prototype such as educational, training, entertainment, information etc. and who benefits from them. The objectives include:

1. Reviewing existing relevant games regarding AMR and agriculture
2. Developing an early prototype game about antibiotic use and livestock production
3. Evaluating the game's representation of AMR functionality and how it could be used

The justification for this aim is that traditional mediums of expression are generally limiting, and I would explore the beneficial outcomes of using modern technology (3, 4). Serious games are advantageous in that they can offer a risk-free environment to play test and explore many different topics. A good case example is Microsoft Flight Simulator where otherwise training can be limiting by high costs and dangers (5).

Since the theme is livestock health management, I set the intended players as farmers and vets. Figuring out an audience that may benefit most from the awareness brought by this prototype is one of the goals I set for myself. The gamified interaction between animals and antibiotics is the initial interest to the prototype. The prototype had been developed iteratively, with in-development prototypes presented to farmers and vets throughout the process and changes and updates the prototype made accordingly. The final prototype is then evaluated with potential users as the method to realize the use of serious games on the topic.

The approaches used in carrying out this project involve:

- Game prototype development through *Unity*
- Requirement collection by means of *User Centered Design*
- Game evaluation and identification of useful direction by *MEEGA+ Model*

The software context involves using Unity Editor, a multiplatform 3D game engine which has built-in components and includes their toolset. With its flexible game space-oriented editing, it would allow for the project to change direction to match changes in requirements easier than other programs. This is supporting the design and development theories emphasizing end-user requirement flexibility over long-term design plans. Therefore, Unity and C# combination had been the context of this project.

The main results of this project point out that farmers are quite interested and happy to engage in gamified tools showing the existence of interest for innovative representations of AMR in agriculture. The game, despite falling under initial skepticism, had been received well by the audience of health professionals and farmers. This research succeeds in finding a gap and providing a well-received innovative serious game, *Where's Woolly*, on AMR practices in agriculture.

2. Background

Before discussing AMR games in detail, it is useful to understand the wider context and advantages serious games have broadly offered. Serious games have recently developed in collaboration with doctors and medical professionals to bring awareness through innovative solutions to communicating the issue and reducing antibiotic use.

a. Serious Games

The criteria of a serious game is divided into three parts being gameplay, purpose, and sector (G/P/S) (1). Gameplay is simply the set of user interactions can take to bring about a state in the game. While traditional games are a form of entertainment, serious games are defined by their targeting a purpose or sector other than that. For example, *Microsoft Flight Simulator* is a serious game with the purpose of teaching the operation of real aircraft (6). *Papers, Please* is a game in which you play a customs inspector to explore the ethical and moral difficulties they may face (7). They have shown to use immersion and realism (8), beyond graphically, to accurately capture their environments and are hence successful games with a higher purpose. Moreover, they can provide the user a 2 or 3-dimensional space to assist visualizing problems better and negotiate any level of detail and engagement appropriate to the problem and audience over traditional learning methods.

Serious games provide the advantage of allowing those environments to be local to the users where they might otherwise have difficulty finding access to an airplane for flight training, as in the case of *Microsoft Flight Simulator*. The risks and costs of failure are therefore reduced by their inconsequential nature, serving as a method of experiential learning (9), and more immersive than traditional education (10). This allows our game to connect non-farmers, or those away from farms, to experience farmers' decision-making process in using antibiotics.

Serious games also bring awareness to the severity of the problems through visual experience and expressive virtual characters (11). *Hazy Days*, a breathing simulator, has you trying to keep a 10-year-old girl in China healthy by controlling her breathing against China's smog problem. Figure 1 shows the expression of the girl changing based on how well the player is performing. The visuals and player experience can impact the player to care more about the topic than traditional education. Experiencing tragedies such as illness or death could be useful motivations in our game for the player to be more attentive and careful over animal health management which integrates awareness well.

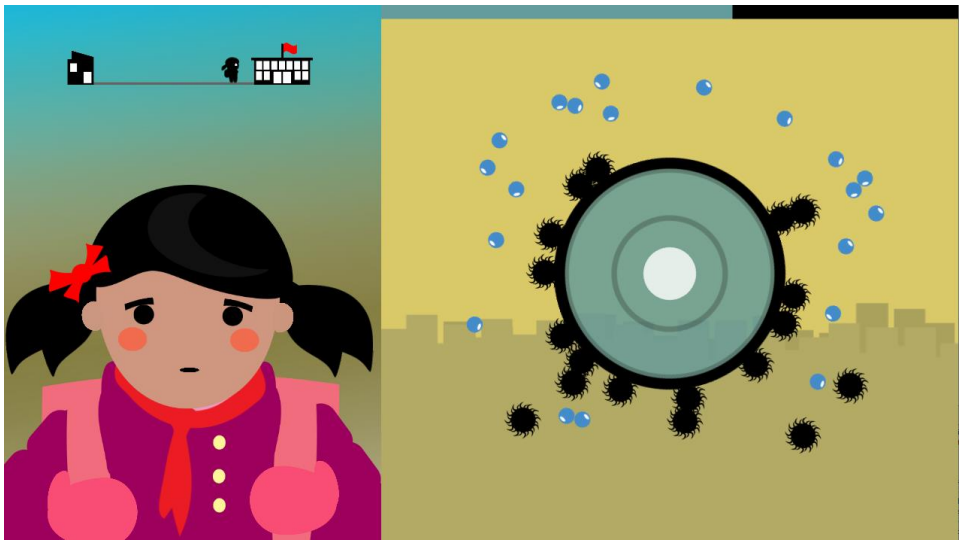


Figure 1: Hazy Days; A serious game about the smog problems in China

b. AMR Games

Serious games have been used by medical professionals as well as AMR scientists for the advantages highlighted in the previous section (12). AMR can be difficult a concept to convey as the effects are invisible and gradual, it benefits from using serious games as a platform of spreading awareness. Generally, AMR games can be broken down into two purposes: basic scientific education (13), and professional level education (14).

Games like *Edugames4all MicrobeQuest* and the *e-Bugs* collection, use cartoonish visuals and simple gamified representations of the problem of AMR (15). This approach is used to make the game more appealing and attractive to children as the target audience. The setting of the game takes place on a school, including the classrooms and yard, as an environment of familiarity to promote that there is more in the immediate surroundings than meets the eye. The game cleverly uses realistic and practical solutions for the children as game mechanics, mainly the importance of one's hygiene such as using tissue paper and washing hands with soap (16). The point from this is that a practical problem-solving tool is important to understand and communicate the available options, for example, farmers would likely use anti-biotics as the means to deal with a sick animal and this would be our main mechanic.

One of the most successful educational mobile games is *Superbugs*, winning the Longitude Prize, whose clever mechanics have enabled it to reach an adult audience also (17). This game has the player dose antibiotics on a petri-dish to stop harmful bacteria growing on it. The trick however is that the longer they last, the more likely they are to become a superbug and immune to players' attempt to use antibiotics to remove them. That is until the game's calendar offers players medical advancements as antibiotic upgrades which can remove the current multiplying superbugs only for the cycle to repeat with the medical upgrade being further away. The game has a strong focus on teaching AMR and assumes players will acknowledge it and act more responsible. I benefited from understanding a good implementation of time pressure's importance as well as penalizing the player based on suboptimal practices as useful design direction for our game. The game also does well to gamify the feeling of impending danger and helplessness the player should experience when trying to dose a superbug with antibiotics (18). Despite its success, one main criticism I have of this game is that the gameplay is abstract and does not teach the player anything about the practicalities of dealing with the problem they or others can apply, as I had seen their importance from before. It is assumed to be successful through popularity; there had not been any analysis on whether the players had changed their habits.

For professionals, I had seen gamified training tools such as *AntibioGame*, where you take the role of a doctor dealing with patient symptoms and deciding when to turn to antibiotics (19). The setting is in a doctor's office, a suitable training environment for the intended audience of medical students. The game has the player give realistic and explicit diagnoses, examinations and prescriptions based on cases written by a medical team to assist the student to revise their accuracy of antibiotic prescriptions motivated by a reward system. This tool also considers students could use a visual break from traditional study and opts for cartoonish visuals. It is not however uncommon for it to look less casual like in the case of *Prognosis: Your Diagnosis* which has similar gameplay. I had seen the importance of environment selection again here but beyond that, I had also noted the explicit and realistic measurements of health monitoring could be useful as a diagnostics tool for our game between the player and the animal due to lacking voice.

AMR in serious games are described well in a clinical scope, with great care put into design choices, but it is not a problem exclusive to them. The animals in countries such as the US consume over twice the medically important antibiotics, stated by (2). Despite this, there are no examples of agriculture-based AMR games to my knowledge.

c. Lack of AMR in Agriculture Games

I investigate the underrepresented key themes in farm games. Extensive researching reveals there are currently no results on games dedicated to AMR in agriculture. I propose the role of farming in antibiotic use is not understood well enough, at least when compared to clinical use, despite being one of the highest industrial users of antibiotics. Through this project's game, there should be more awareness and advocation for responsible use of antibiotics focused on farms.

Another reason is that farm games drift towards idyllic rurality (20) due to their selling point that they are natural and peaceful environments through sanitized visuals and gameplay. This includes a selection of well-known games such as *FarmVille*, *Stardew Valley*, and *Farming Simulator*, in which animals are entirely devoid of the concept of disease. Instead animals are treated as a means of harvesting resources without the need to moderate their health and wellbeing, which may even reinforce unimportance of their health to a harmful extent. This project prioritizes representing these lacking concepts of disease afflicted animals and formulating main gameplay around them.

The lack of animal illnesses is especially surprising in *Farming Simulator*, which is a hyper realistic farming simulator. Due to its level of realism, a significant proportion of its audience and feedback team as farmers and farming students (21) yet it does not attempt to portray any medical practice, reaffirming my proposition that there is lack of both awareness and integration to serious games. I had mentioned animal health representation being a priority but another important point to take from this game is hyper realism has attracted a farmer player-base and development involvement which are both important to this project's game.

d. Summary

It is apparent that there are many advantages of using serious games as a platform for education, training, and communication purposes as opposed to the traditional formats. Some of these innovative capabilities of games included: providing immersive environments to those out of reach, the risk- and cost-free aspects of experience-based learning, adding a sense of responsibility through player immersion, and communicating practical solutions through themes and gameplay mechanics. Many modern gamification techniques have been used to spread awareness of AMR such as the clinical dilemma of prescribing antibiotics, the pressures of time on disease intervention, their visual design decisions to relieve students from the nature of a work environment, explicit communication of symptoms patients exhibit, motivational reward systems, and the helpless feeling dealing with highly resistant microbes. I believe however, there should be more of a highlight on practical solutions represented in each industry to communicate good practice through serious games.

Current games are heavily learning towards raising awareness of the overuse antibiotics in a human health setting, despite the significant industrial use of antibiotics on farms. There is clearly more that can be done regarding AMR in agriculture through innovative awareness platforms, in our case, serious games. There are many farm and AMR games to look to for inspiration, but this project requires research through farmer engagement, as the audience, on their animal health management practices to better understand them first. Combining this acquired knowledge and serious game techniques, the aim of this project is to build a game that satisfies the need of spreading awareness on reducing the use of antibiotics through pragmatic game mechanics.

3. Specifications & Design

This project is carried out in collaboration with researchers with specialized expertise on the topic. Therefore after initial specifications gathered through field research, there had been a set of final requirements suggested by the team.

This section is broken down into:

- Justifying simplistic design of a primitive prototype based on characteristics of similar serious games
- Our overarching iterative methodology for meeting user requirements: *User Centered Design*
- Using *Thematic Analysis* to gather requirements from feedback centric design
- Build a *Requirements* list based on feedback themes
- Expert suggested requirements for personalization
- Use *Use Cases* and interactions with the program
- *Gameplay Design* choices that resulted from feedback and requirements

a. The First Prototype

I had started with the wireframe, seen in Figure 2, as a tool to communicate between me and the team on understanding how they envisioned mechanics of an AMR farm game. This led to a lot of discussion and misalignment on expected specifications and particular design of gameplay. However, this wireframe discussion led to some useful requirements coming out and me to some more insights about the difficulties of diagnosis on animals. I also better understood the differences between severity and antibiotic resistance and thought this would be helpful to keep a note of.

Figure 2: Image of the wireframe

For our initial prototype, only top-level requirements had been used to speed up development to reach user feedback stages sooner. This is because both on design and development, I use feedback-oriented approaches and therefore want to suggest the core mechanics of our prototype to our end-user to turn ideas and criticisms into useful requirements.

The Functional Requirements are:

- The game must contain an interactable set of animals,
- The player must have the option to give an animal medication,
- The game must display text of an animal's conditions and symptoms upon selection,
- The game must give feedback to the player on their practice upon request,
- The game must have replayability through randomization.

The Non-Functional Requirements are:

- The game must recognizably portray a farm,
- The player must clearly be able to see and recognize all animals,
- The game text font must be readable (size and colour).

In our prototype, Figure 3, the core mechanic of gameplay is determining whether an antibiotic should be dosed to an individual animal. Cows had been initially selected as the game animal due to most interviewees having worked with cattle. Thus, the player should take the role of a farmer and freely medicate animals he/she deems ill. A 'journal' entry, as seen in the top left of Figure 3, is a list of animal symptoms that is available to the player upon selecting the animal, an idea borrowed from the AMR games discussed previously. The symptoms however are arbitrary and are not connected to whether an animal is ill. This is because I am currently uninformed on farmers' diagnosis strategies.

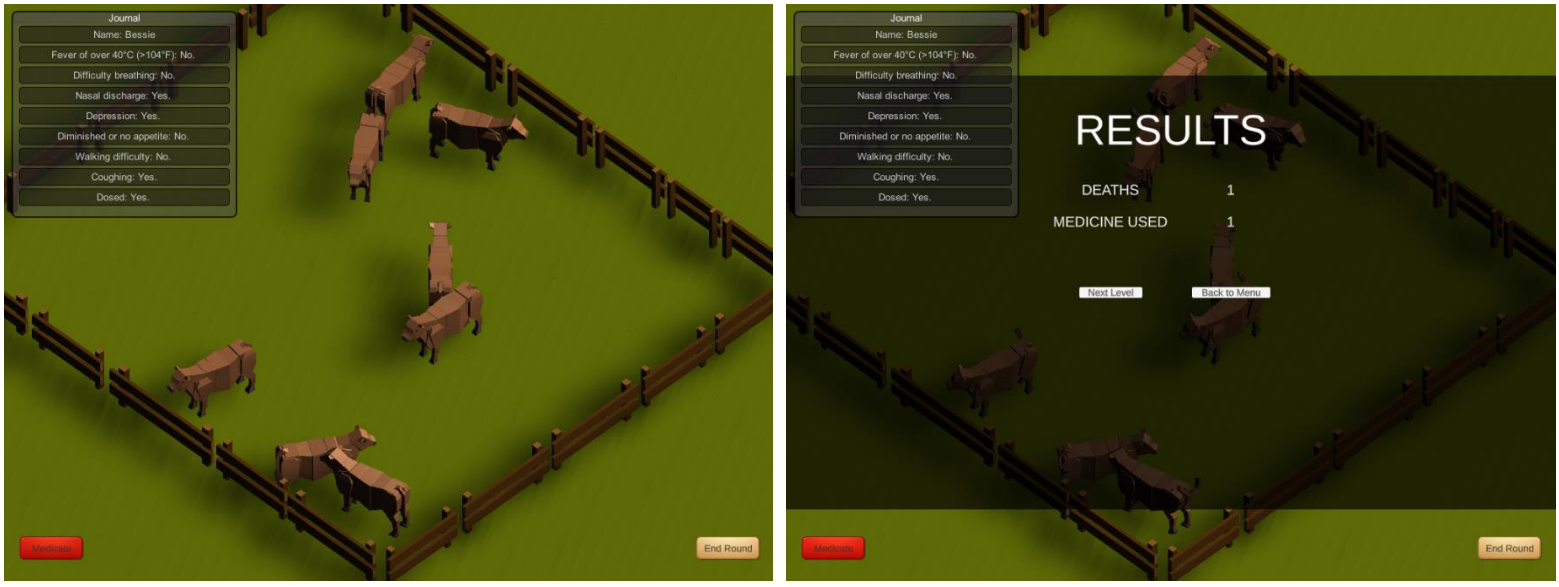


Figure 3: Initial and basic prototype built from early requirements

b. User Centered Design

User centered design (UCD) is an iterative design approach that focuses on user feedback to help software meet end-user needs (22). It places emphasis on maximizing usefulness by suggested requirements made by the potential users and benefits by working towards practical applications. Because this is the first AMR game regarding farm animals, I needed constant constructive feedback for design to be appropriate and useful. The team had got in touch with farmers through JustFarmers website and had been in contact with others previously. The prototype is presented to farmers during interviews to receive constructive criticism and further ideas for game requirements. After each set of interviews, I return to the prototype, reassess design and also update the prototype. Once the changes satisfy the feedback, I present our refined prototype to a new set of interviewees for a new set of feedback. This cycle can be seen in Figure 4. This method also ensures our work was constantly validated by experts as to not mis-construe a point.

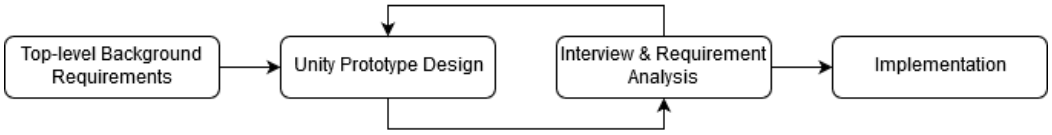


Figure 4: User Centered Design process

I pair 2 interviews per week per version of our prototype. I also receive iterative feedback from the expert involved. For example, I would interview 2 farmers/vets in a week, do analysis on their feedback and criticisms, and use the rest of that week to update design and implement changes to build prototype version 2. I would then repeat this the next week but instead present prototype version 2 to two other farmers/vets. By iterative methods, the goal is to push our prototype to meet end-users desired interactions and UCD to work towards meeting usability requirements of the industry.

c. Thematic Analysis of Farmer Engagement

Thematic analysis is a qualitative based analytic method which I used to gather requirements through interviews and feedback sessions with farmers and vets (23). It is a tool that allows us to identify patterns in a dataset such as the ideas, opinions and perceptions an individual has on a topic. For this project, this would be useful to identify common trends in desire and expectation from our prototype. I identify "codes" in the datasets which are short labels to add up to common themes and ideas.

I interviewed farmers and vets regarding our topic and hope to find common trends among their medication practices with animals and expectations of functionality regarding the serious game. Our interview is split into two sections: practical questions regarding farm operation and prototype feedback. The responses had been translated into either new design choices, or updating current designs to better suit user requirements. The interviews are long however, so the few more important and critical excerpts are addressed here.

i. Ethical Approval

I had applied for ethical approval for the interviews, interactions and data collection of the farmers and vets. This included recording the interview for reflection. My certification is attached as Appendix A. I had written up a set interview questions as a study protocol and have attached them as Appendix B.

ii. Week 1: Interviews

Context	Interview Extracts	Codes
Discussion session with Interviewee 1: Asked about method(s) of spotting signs of disease.	"It's stockman-ship, so that is spotting the abnormal. So good farm stockmen and vets are able to tell when an animal is in its normal behaviour and if it's doing something abnormal then it's the first flag that something is wrong. So perhaps in a bunch of animals in the pasture, one is on its own. Well they tend to stay as a group. So if it's on its own, it's struggling to keep up with its mates for some reason. Or if it's in a shed, it's not feeding as much as the others… so when fresh food is put up, it doesn't come and eat it. It makes sense to spot them early because your better chance of a successful intervention is earlier in the disease process. Good stockmen spot them early, bad stockmen discover them dead."	• Vets and stockmen used interchangeably • Reading behavioural signs with intuition • Early intervention linked with success • Animal life dependent on stockwork
Feedback on prototype version 1 by Interviewee 1: General suggestions of improvement of the prototype.	"More activity. Things which I guess are really difficult so making it look like a farm, making the sheep look more like sheep, having them wandering around grazing, going to have a drink and feed and the ones that aren't doing that regularly then you could pick up the subtle things. If we were trying to teach stockman-ship to farmers, we would almost have to have is real animals with real signs on a real farm either as video or actually being on a farm."	• Behaving like real animals to identify subtle changes in behaviour • Emphasizing video level realism of natural surroundings and animals
Discussion session with Interviewee 2: Asked about method(s) of spotting signs of disease.	"The thing is you develop what is known as "Stockman's eye". You develop that through your life. You know if your cow or sheep is not right. Because she will look at you in a certain way or make some motion that's not right. So then that is the time when you intervene."	• An experientially developed intuition • Standout motions and behaviour • Intervention on signs being read
Feedback on prototype version 1 by Interviewee 2: Visual suggestions of improvement of the prototype.	"A bit more field furniture. A hedge, a couple of oak trees or an ash tree. More environmentally friendly. I'll tell you what it looks like now, it looks like a child's pretend farm. Trees are not only lovely scratching posts but they're also lovely shelter."	• Natural environment decor • Unrealistic visuals can be distracting • Environment interaction by animals

Table 1: Week 1 interview extracts & codes

iii. Week 1: Thematic Analysis

Codes	Themes
○ Vets and stockmen used interchangeably ○ Animal life depends on stockwork ○ An experientially developed intuition	Stockman Impact
○ Reading behavioural signs with intuition ○ Behaving like real animals to identify subtle changes in behaviour ○ Standout motions and behaviour ○ Environment interaction by animals	Animal Behaviour
○ Intervention on signs being read ○ Early intervention linked with success	Timely Intervention
○ Emphasizing video level realism of natural surroundings and animals ○ Unrealistic visuals can be distracting ○ Natural environment decor	Environmental Realism

Table 2: Week 1 codes & themes

iv. Week 1: Evaluation & Requirement Extrapolation

I had gained a lot from the first iteration of feedback. I apply thematic analysis on some week 1 extracts, Table 1, and arrive to a few important themes as in Table 2. I heard a new term that caught my interest: "Stockman". A Stockman is a person responsible for caring and tending to livestock needs, including catering to their health. I had learned of the importance of Stockmen in animal health management through thematic analysis. Stockmen rely on qualitative signs and animal behaviour as opposed to our quantitative set of communicated animal symptoms in the prototype. Based on various animal signals, a Stockman would hope to pass earlier judgment for successful interventions, highlighting importance on a time factor. Our environment was also noted not be sufficient and authenticity and realism is a key expectation from the product.

Functional Requirements

Requirement	The game must only use visual behaviour to communicate illness to the player
Rationale	To portray the role of stockmanship accurately, the only information the player should receive on the animals is visual cues. This is achieved by the system controlling animal animations to portray ill behaviour.
Requirement	The game must have more than one representation of odd behaviour
Rationale	To capture stockmanship accurately, the player is expected to look for various signs of illness and not limit it to one type of behaviour. There should be multiple level types with different behavioural cues.
Requirement	The game must include time pressure
Rationale	To emphasize speed as an important factor of intervention, there should be time related pressure. The player must be penalized for taking too long in increments.

Non-Functional Requirements

Requirement	The game should provide a challenge
Rationale	To communicating to the player that animal behavioural cues are subtle, but not impossible nor obvious. Behavioural mechanics will be tuned to a satisfactory subtlety.
Requirement	The game must contain high resolution and realistic textured animals
Rationale	To meet our farmers demands for a product of a serious nature, the visuals must be video-like and emphasize realism. Therefore, animal and environment models must be realistic looking.
Requirement	To game could include natural environment décor
Rationale	To immerse player into farm environment, there could be some natural outdoor objects. The game could include some bushes, hedges, trees, drystone walls etc. into our level layout.

v. Week 2: Interviews

Context	Interview Extracts	Codes
Feedback on prototype version 2 by Interviewee 3: Asked to identify a sick animal in our game. The sick animal was distancing itself from the herd.	"Probably the one standing on the right there on its own but it's hard to tell from the view. The ewe with the lamb under her chin is doing a good job. The ewe is flicking her ear? I didn't realize she could flick her ear. And her tail, she's mucky tail. I don't know what time of year we're on, we could be looking at maggots there. There's flies about cause they're flicking their ear."	• Camera Viewpoint • Seasons dependent diagnosing process- • Animations cluing many signals-
Feedback on prototype version 2 by Interviewee 3: When asked about how it could compliment on-farm use.	"Yes, it would support on farm learning cause it could be the testing of what people have learnt. I think you should have more videos as well, then perhaps go to the animation because farmers love to see other farms. So they'll see all the things you don't want to see like gosh why is that gate not hanging properly, or why hasn't he thought to move the water truck somewhere else. If you put something realistic and then you put the animation that will really test him because the animation isn't easy to spot. It's got to challenge the farmer and it's got to be entertaining. I could see a role for it but you would need to have some realistic shots as well. Farmers love to look at stock and even more they love to look at tractors."	• Reinforcement learning- • Environment structures and machinery for appeasement- • Testing through a challenge-

vi. Week 2: Interviews (Cont.)

Discussion session with Interviewee 4: Asked about method(s) of spotting signs of disease.	"We get lame animals, hmm, they're mostly in large groups so it's difficult to spot them when they're off their food. When the sows, the adult animals, are on their own in farrowing section, you can easily spot that they're not well. They don't eat basically. Maybe they might have distended utter or you can spot their piglets aren't doing well, they might be scouring or getting thin. She might have something wrong with the udder or something. The other way we monitor their health, because we run them in big groups, is if the performance of the group deteriorates, we can often spot that if they're coughing or they don't appear to grow very well or the mortality of the group is high compared to where we think it ought to be, or there are a lot of plain looking animals. If you look at a domestic dog, they're looking healthy but a stray dog a bit rangey."	• Professionals struggle with diagnosing at times- • Sickly physical characteristics-
Feedback on prototype version 2 by Interviewee 4: When asked about how it could compliment on-farm use.	"We occassionally have antibiotic refresher meetings with the staff and something like that for them would be quite useful and fun, to present it like that, in other species as well. Especially from a training point of view as a training tool to spot the various things the animals do that are different to the other animals. That would be very useful. A young person coming to the industry, they've got to learn quite a lot, it's not easy looking at animals if you've not trained yourself to look at them over time. This might be a short circuiting experience which is what training is all about."	• Branching animal types with behaviours- • Amused by representation and sharing- • Shorten training-

Table 3: Week 2 interview extracts & codes

vii. Week 2: Thematic Analysis

Codes	Themes
○ Reinforcement learning ○ Testing through challenging ○ Shorten training ○ Professionals struggle with diagnosing at times ○ Amused by representation and sharing	Training and Revising
○ Branching animal types with behaviours ○ Sickly physical characters ○ Seasons dependent diagnosis process ○ Animations cluing many signals	Variety and Depth
○ Camera Viewpoint ○ Environment structures and machinery for appeasement	On-site View

Table 4: Week 2 codes & themes

viii. Week 2: Evaluation & Requirement Extrapolation

I had gained some insight on game structure for potential uses from this set of interviews, especially the quotes on Table 3. Through the themes of Table 4, farmers suggested ways in which the prototype could be used for revision as well as training. The innovative teaching method could lend itself useful to trainees and professionals alike as the job requires accumulating and refreshing one's knowledge. This had been applied through different game modes, one for testing and one for practice, presentation, or revision. They had also described aspects I had not thought of such as the time of year, the game animals making extremely subtle movements, sickly physical characteristics and reaffirmed some I had considered such as having more animals. Having more animals for example, is more desirable but I am interested in understanding the expected gameplay from a single animal before the game covers a broader variety of animals. I put some of these on the backlog due to time and resource constraints, at the end of the project paper.

Functional Requirements

Requirement	The game must have an assessed play game mode
Rationale	To motivate attentive and good practice by the players, there must be a game mode which has stakes and rewards. This can be achieved by having consecutive levels which captures overall performance and provides feedback.
Requirement	The game should allow playing individual behaviour levels
Rationale	To understanding and revise the individual behaviours better, the player can practice identifying the individual symptoms. A dedicated scene will be used as a menu to select various illness cases to access with simpler mechanics.
Requirement	The game must include menu navigation
Rationale	To give players choice of accessing game modes, options of play, settings or even quitting, there should be a homepage style scene. There will be a dedicated scene the player starts on and can return to which contains all game options.
Requirement	The game must have a numerical scoring system
Rationale	To assess good practice and reward it, the game needs to motivate players to be attentive of their choices by giving them feedback. A numerical score will be given to players based on their speed and accuracy.

Non-functional Requirements

Requirement	The player should have a clear viewpoint
Rationale	To have the animals on scene clearly visible altogether with their animations visible to the player. A clearer overhead and isometric camera view will be implemented.
Requirement	The game could provide a farm feeling locale
Rationale	To add a little more immersion, the game could add some man-made farm tools, machinery, and structures. I could add more environment and object assets from the Unity asset store which are farming related.

d. Team's Requirements

This phase involved showing the latest prototype to collect the expert final requirements and system expectations. The team had listed the following final requirements to add:

The Functional requirements are:

- The game must have medication interaction feedback,
- The game must feature losing,
- The game should display accuracy as part of feedback,
- The game must have a scoring system related to accuracy,
- The game must have instructions to the rules of the game.

The Non-functional requirements are:

- The game should feature visual impact of losing,
- The game should feature ambient background sounds.

e. Realism Attraction

There were many requests and points of discussion regarding perceptive realisms. Interviewees mentioned needing realistic animals and realistic environments, including flora and related farm themes. Some even mentioned incorporating actual videos of farms. It is also seen before that farmers seem to be attracted to the realism in the farming simulator series. The team also mentioned auditory realism. I had concluded realism is an important design element which keeps players immersed, focused, and interested. The project had scrapped the initial assets it used for this reason and then had invested in acquiring appropriate assets that meet user requirements for the purpose of building in game environments.

f. Use Cases

The Use Cases of Figure 5 in this section describes the interactions the player has with the software in order to execute and experience the various features available. There are four main functionalities under the users control:

- Freeplay, Table 5, allows the player to play a set of levels and get feedback by the end.
- Cases, Table 6, allows the player to explore and play the individual levels separately.
- How to play, Table 7, displays the instructions of Freeplay mode.
- Settings, Table 8, allows the users to customize a few parameters to their preference.

Case Name	Playing Freeplay
Case Description	The player can use the software to play the main format of the game and receive feedback by the end of it.
Pre Conditions	- Windows 7 SP1 or any later version - Graphics card with DirectX 10 or any later version - The program file has been downloaded
Post Conditions	- The player played the freeplay mode
Basic Flow	1. The player runs the executable to open the software 2. The player selects the “Freeplay” option from the main menu 3. The player plays the levels until they are complete 4. The player receives feedback 5. The player is returned to the main menu
Alt. Flow	1. The player returns to the main menu from other states 2. The player selects “Freeplay” from the main menu 3. The player is playing the levels 4. The player exits or fails back to the main menu before the levels are complete

Table 5: Use Case – Playing Freeplay

Case Name	Playing Cases
Case Description	The player can navigate the software to the Cases menu and select individual cases. The player can then select a case and play it individually.
Pre Conditions	- Windows 7 SP1 or any later version - Graphics card with DirectX 10 or any later version - The program file has been downloaded
Post Conditions	- The player is played any of the cases
Basic Flow	1. The player runs the executable to open the software 2. The player clicks the “Cases” button on the main menu 3. The player selects “Case 1” 4. The player plays the individual level/case
Alt. Flow	1. The player returns to the main menu from other states 2. The player clicks the “Cases” button on the main menu 3. The player selects “Case 2”, “Case 3”, or any other case 4. The player plays the individual level/case

Table 6: Use Case – Playing Cases

N.B., The use of “Case One”, “Case Two”, etc. To represent cases arbitrarily as options. They are not limited to exactly to three or phrased as such.

Case Name	How to Play Instructions
Case Description	The player can view the rules of the freeplay mode.
Pre Conditions	- Windows 7 SP1 or any later version - Graphics card with DirectX 10 or any later version - The program file has been downloaded
Post Conditions	- The player views/understands the rules of the game
Basic Flow	1. The player runs the executable to open the software 2. The player clicks “How to Play” and reads the instructions
Alt. Flow	1. The player returns to the main menu from other states 2. The player clicks “How to Play” and reads the instructions

Table 7: Use Case – How to Play

Case Name	Change Settings
Case Description	The player can access some technical settings and change them.
Pre Conditions	- Windows 7 SP1 or any later version - Graphics card with DirectX 10 or any later version - The program file has been downloaded
Post Conditions	- The player had set preferred settings
Basic Flow	1. The player runs the executable to open the software 2. The player clicks the gear icon in the main menu 3. The player changes the resolution 4. The player returns to the main menu
Alt. Flow	1. The player has the software open 2. The player clicks the gear icon in the “Cases” menu 3. The player sets his/her preferred settings 4. The player returns to the cases menu

Table 8: Use Case – Change Settings

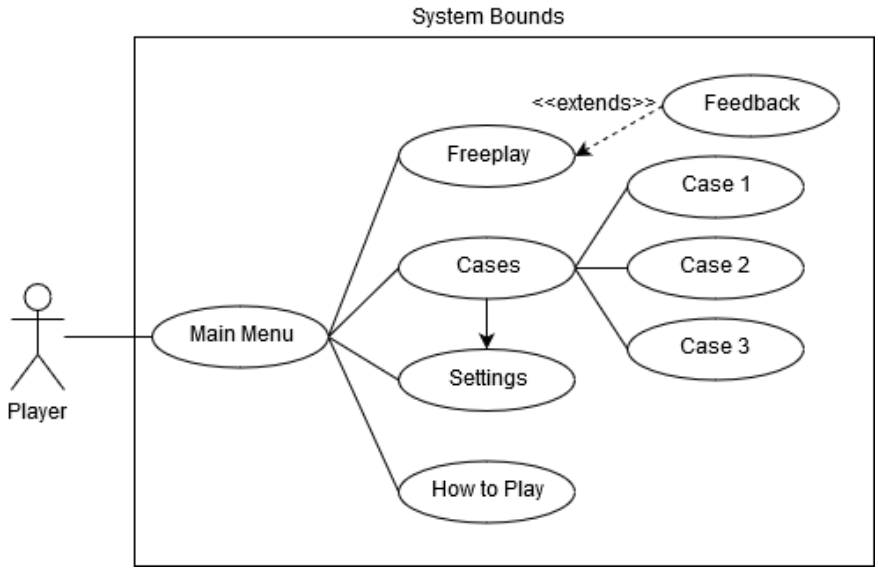


Figure 5: Use Case Diagram

g. Gameplay Design

Gameplay Loop

> *"In Halo 1, there was maybe 30 seconds of fun that happened over and over and over again, so if you can get 30 seconds of fun, you can pretty much stretch that out to be an entire game. Encountering a bunch of guys, melee attacking one of them before they were aware, throwing a grenade into a group of other guys, and then cleaning up the stragglers before they could surround you. And so you can have all the great graphics, and all the different characters, and lots of different weapons with amazing effects, but if you don't nail that 30 seconds, you're not gonna have a great game."*
>
> *- **Jaime Griesemer, Bungie***

The quote above describes the importance of the "Gameplay Loop" which is a collection of actions a player has access to over a short period of time that repeat itself over the course of the game (24). Every game consisting of game actions can be broken down into a gameplay loop. The goal was to come up with a simple design for this loop to how each level is played.

Starting with each level having a common overarching theme; to spot the animal behaving oddly. This approach already resembles the *Where's Waldo* series. Instead of characteristics of the object I am looking for, I use animation as well as other approaches to differentiate behaviours in the game. An animal that's behaving different to the rest is showing symptoms of illness. The player is tasked with identifying the single animal exhibiting symptoms of illness and curing them. The gameplay loop is therefore picking out the sick animal from a herd in order to cure them before moving onto the next, represented by Figure 6.

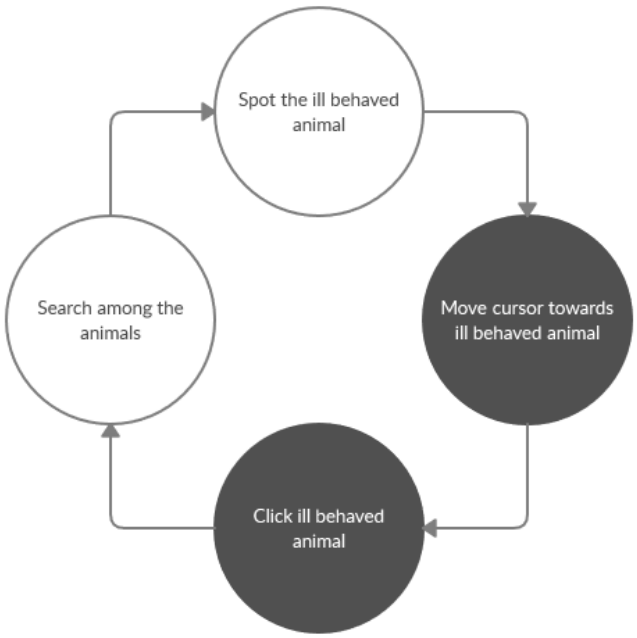


Figure 6: Gameplay Loop

h. Case Behaviours

Coming from interviews, I had determined to make the context of the player playing the role of a stockman. Stockmen rely on different subtle behavioural signs which are different symptoms of illnesses. During the interviews, there were some commonly state behaviours:

1. Distancing themselves from the rest of the herd and self isolating,
2. Not keeping up with the rest of the herd,
3. Refusal to eat fresh food or graze.

The game therefore prioritized to support these behaviours. I had developed functionality for three different sets such that these behaviours would be subtle but apparent among a group of 'normal' animals. The gameplay loop is used as means of interaction and play with each set.

Each scene would spawn 10 animals randomly arranged on a grassy terrain with an overhead isometric camera which serves as player point of view and…

- For the case of an isolating animal, one animal simply spawns at a distance from the rest of the group. The player is expected to notice that one animal is not grouped with the rest and is at a visually measurable distance from the others. To further clue in players on their self-isolation, the animal is made to lie down by itself after a certain period.

- For the case of a lame animal, an animal that is not keeping up with its herd, I had the herd spawn and move with 'flock-like' behaviour. This means animals are grouped up and attached to one another and move as a unit unless they are separated by some radius. Typically, the animal that is separated stops following the herd and decides to stand idly. The ill animal has two different characteristics in this case, it runs as a slower pace than the other animals, more likely to separate, and it has a different animation for its running, a slower looking trot. The animal that is ill is likely to be separated and stand idly until it gets pushed and grouped into the herd only to stumbly run. There are cases where a healthy animal is separated because it does not catch up to the herd and remains idle fooling the player into thinking they are ill. The player must pay attention to the differences astutely in this case as there is a lot of motion and could provide challenging.

- For the case of refusal to graze, the animals sit idly until some decide to eat. The ill animal, however, will never decide to eat. This is carried out by the animals in this scene having rhythmic eating patterned animations, varying by speed to attract the user's attention and add a challenge. The player must identify the animal never grazing as the ill animal.

These case behaviours would be available to test individually and part of the main assessed game mode, called Freeplay.

i. Assessed Play Design

To hybridize play and test for a game mode, somewhat like *Duolingo* (25), would be an appropriate approach. This fits all the requirements of an assessed play mode, providing a challenge, a score and giving accuracy feedback. The designed mode is called Freeplay. Freeplay mode consists of 9 case behaviours, which I refer to as levels. The levels cycle in the order, not-eating, distancing, then flocking, then repeating three times totaling 9 levels as Figure 7 shows.

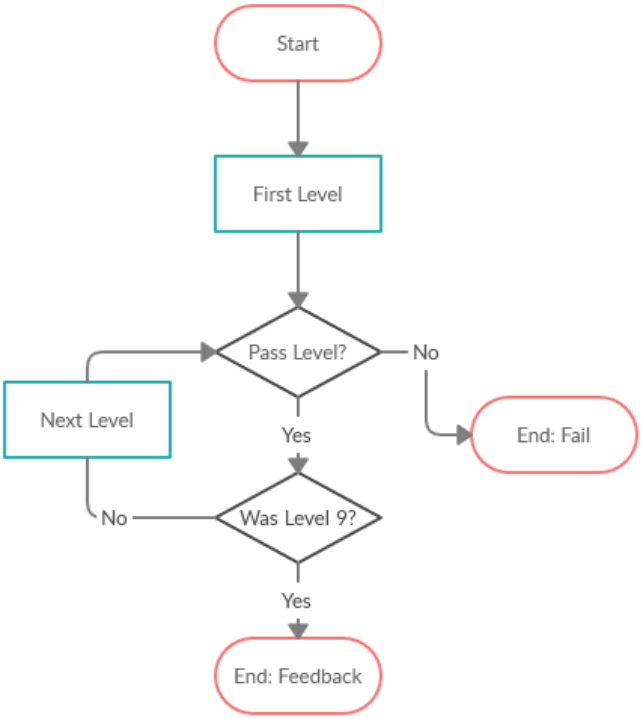


Figure 7: Assessed play activity diagram

To break down Figure 8, players first start with a set of animals and a timer of 20 seconds. The timer ticks downwards every second. The player interacts with animals via gameplay loop until player reaches either 'end' result.

If the player spots a healthy animal and tries to cure it, the antibiotics used will increase by 1 and the game cycle will continue. If the player finds the sick animal and cures it, antibiotics used is incremented according to how long the player took, then the player is taken to the next level's start. If the player was on the 9th level, the player is instead taken to the results screen with feedback results displayed, including their previous best score.

If the player, however, allows the time to expire before interacting with the sick animal, it will play its death animation to let the player know the failure state has been reached. Following that, the player will be removed from the level scene back to the main menu and have his current progress score erased. The player may also click the "back to menu" button to skip back to the main menu at any point during each level.

The user's best score is saved on the main menu as a motivational reminder to try and beat it.

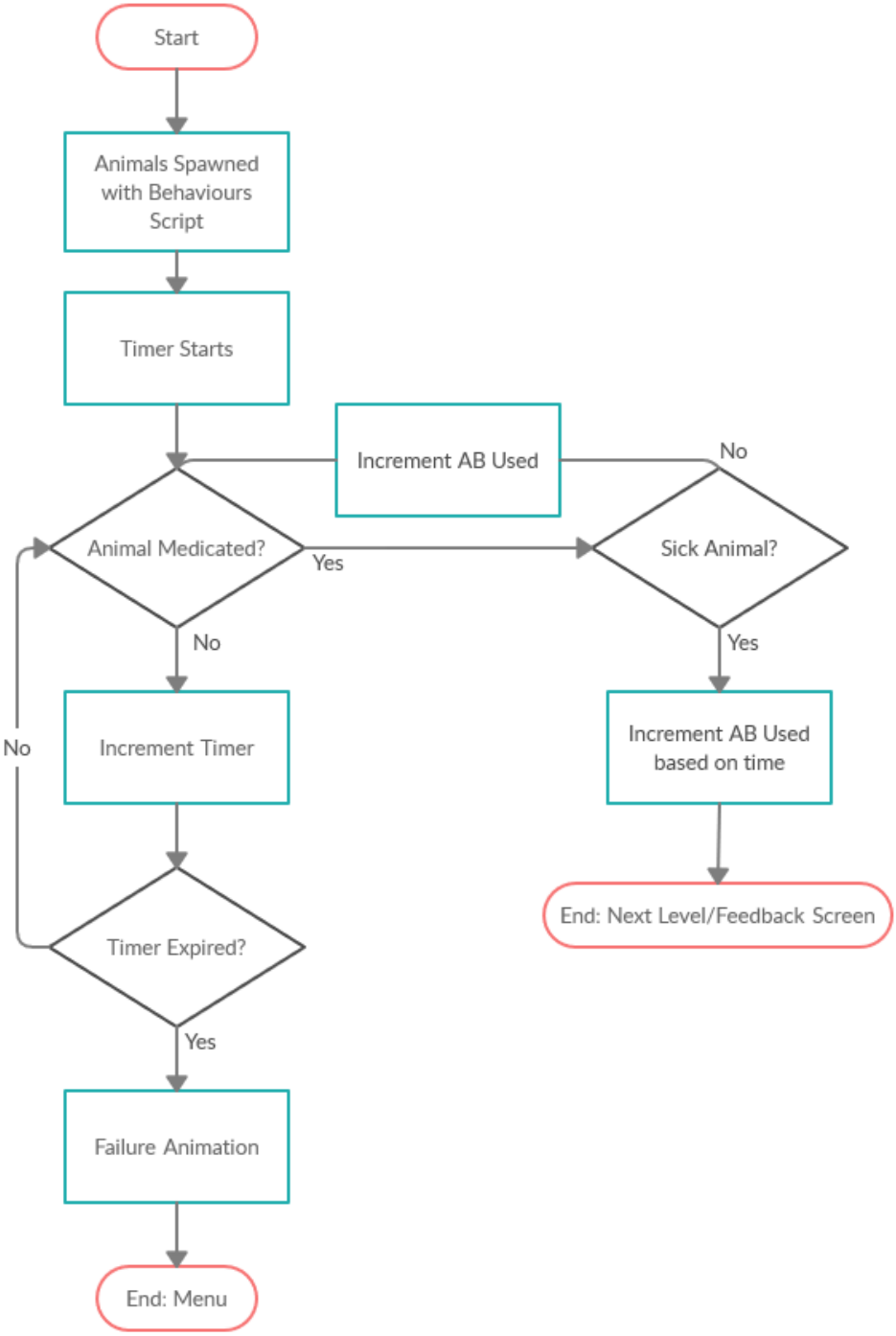


Figure 8: Designed level activity diagram

j. User Interface Design

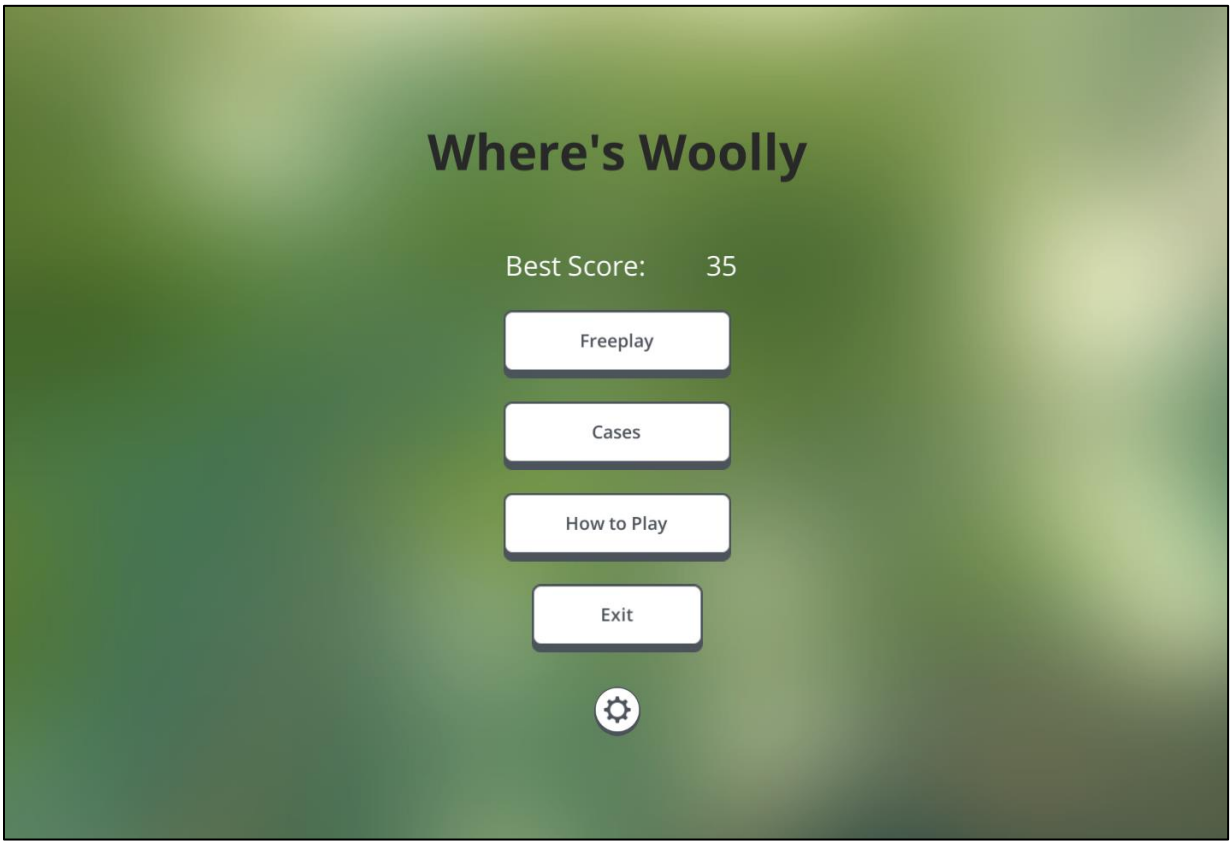


Figure 9: Screen - Main menu

Figure 9 - Features (top to bottom):

- **Best Score**: The best score displays the users best attempt at the 'Freeplay' category. This is used for motivation to set reminder to the player to beat their best score.
- **Freeplay**: This button takes the user to the Game screen of the assessed game mode. Hovering over it describes Freeplay mode.
- **Cases**: This button takes the user to the Cases menu where players can select and play individual cases. Hovering over it describes the Cases menu purpose.
- **How to Play:** This button takes user to the Instructions screen. Hovering over it describes it teaches you how the game works.
- **Exit:** Closes the game. Useful if user is in full screen mode.
- **Gear icon:** Takes the user to the Settings menu.

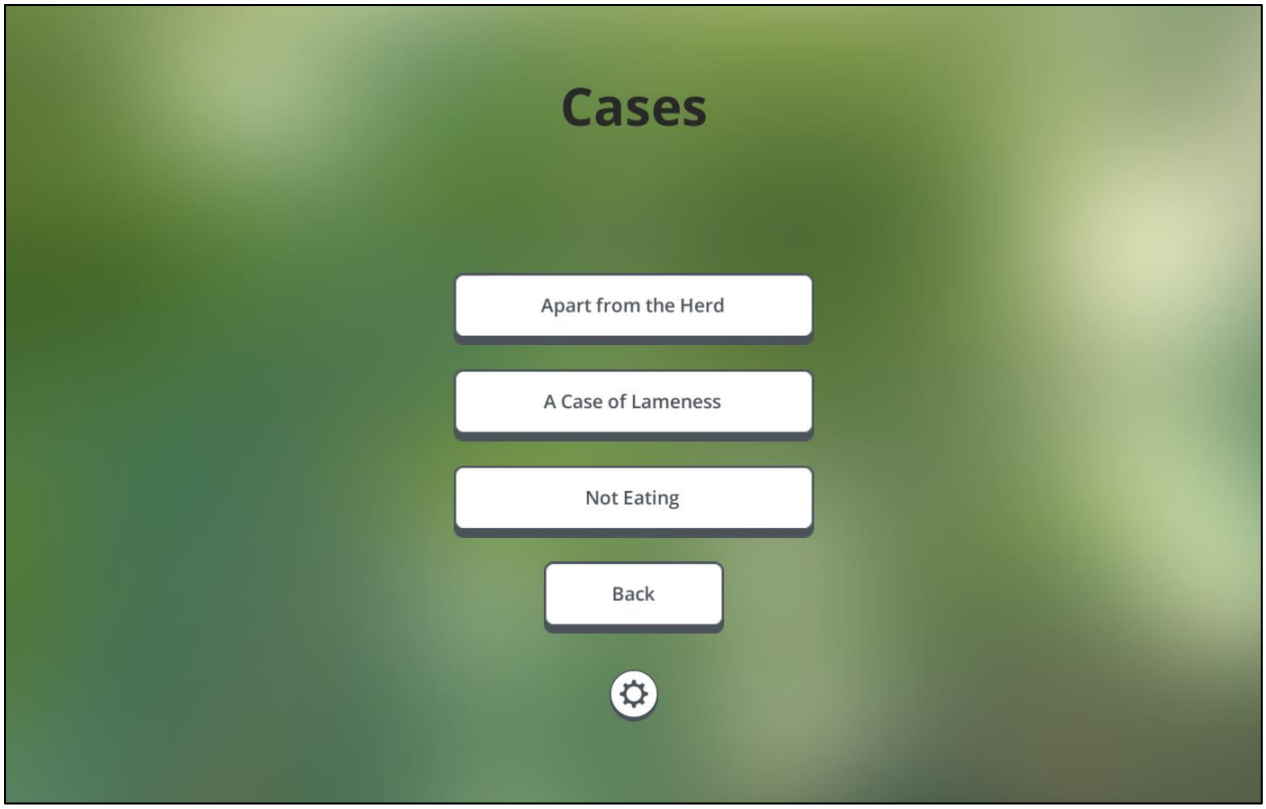


Figure 10: Screen - Cases menu

Figure 10 – Features (top to bottom):

- **Apart from the Herd:** This button takes the user to the practice the distance level Game screen. Hovering over it describes the distance level behaviour.
- **A Case of Lameness:** This button takes the user to practice the running level Game screen. Hovering over it describes the running level behaviour.
- **Not Eating:** This button takes the user to practice the not-eating level Game screen. Hovering over it describes the distance not-eating behaviour.
- **Back:** This button takes the user back to the Main menu.
- **Gear icon:** This button takes the user to the Settings menu.

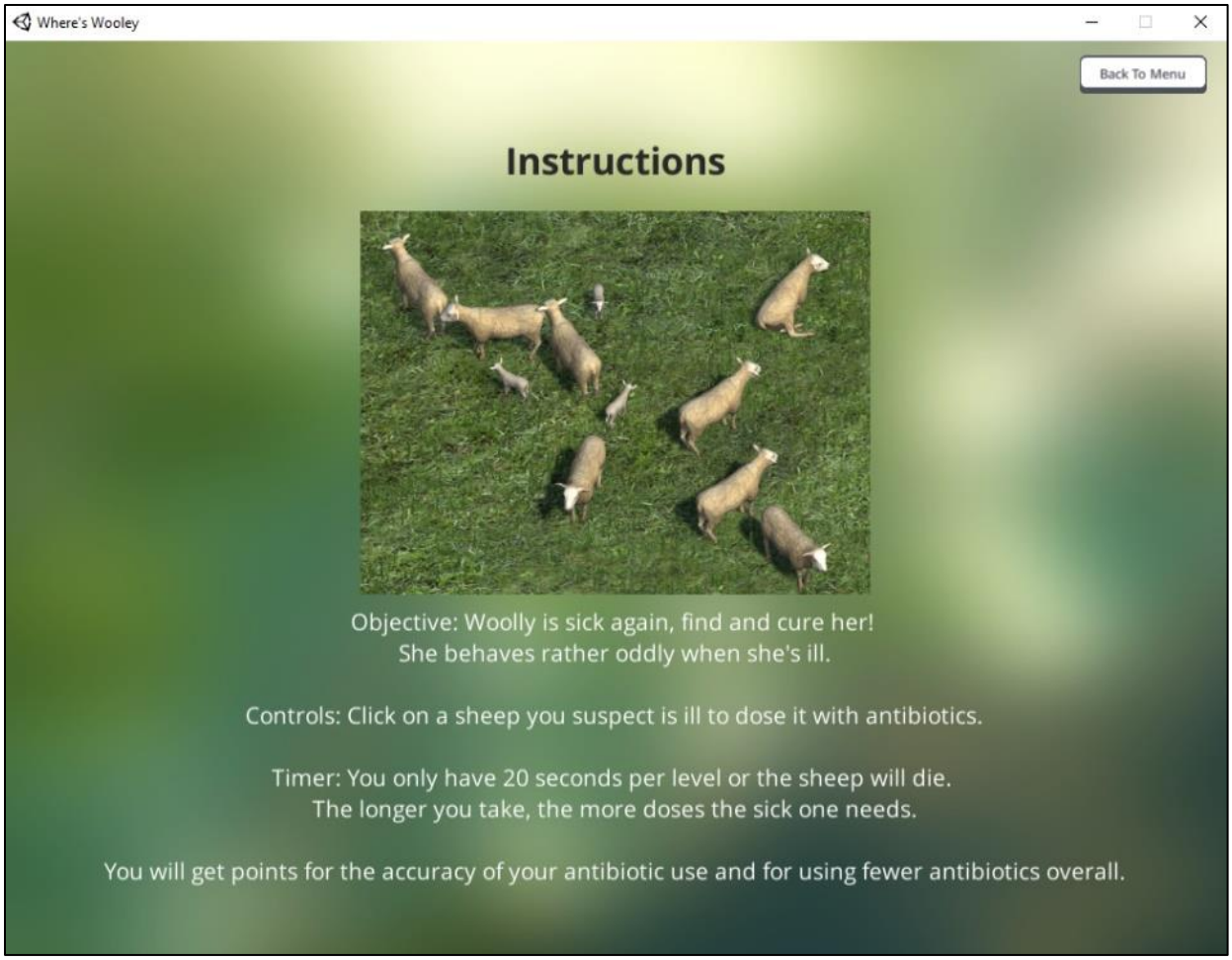


Figure 11: Screen – How to Play

Figure 11 – Features (top to bottom):

- **Back to Menu:** Returns the user to the Main menu.
- **Image:** This is an image of how the game looks. The scene gives an idea of how the game looks to the player along with instructions on how to interact with the given scene.
- **Instruction Text:** These are the rules of the game. This text describes the objectives of the game, the controls of the game, the timer element and its role, and the scoring determinants. This is helpful to the player if they are new to the game or do not understand how the game works.

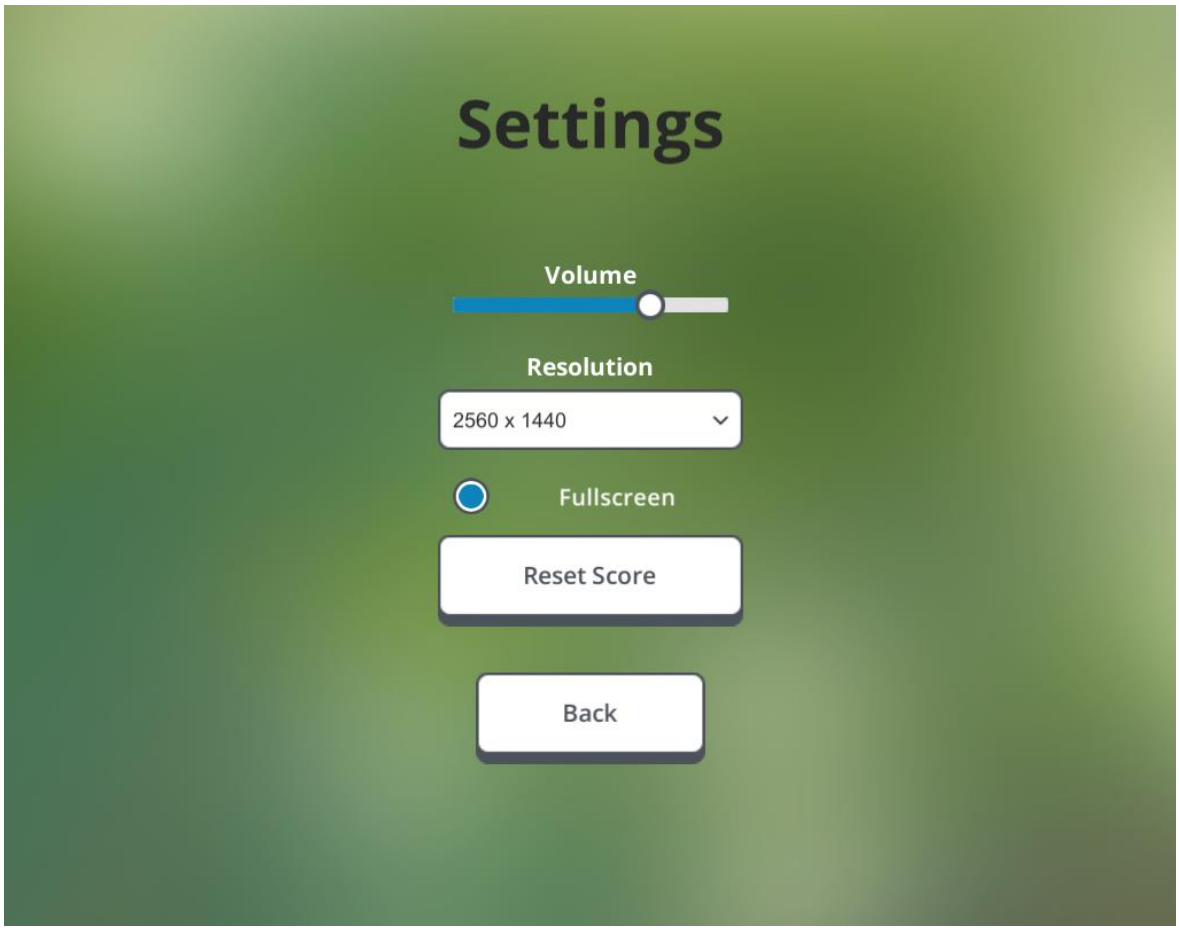


Figure 12: Screen - Settings

Figure 12 – Features (top to bottom):

- **Volume:** This slider allows user to manipulate volume settings.
- **Resolution:** This dropdown allows the user to specify their own resolution if they would prefer.
- **Fullscreen toggle:** This toggle allows the user to switch between full screen and windowed mode to suit their preference.
- **Reset Score:** This button allows the user to clear the current score. The user may want to start a new challenge.
- **Back:** This button allows the user to return to the main menu.

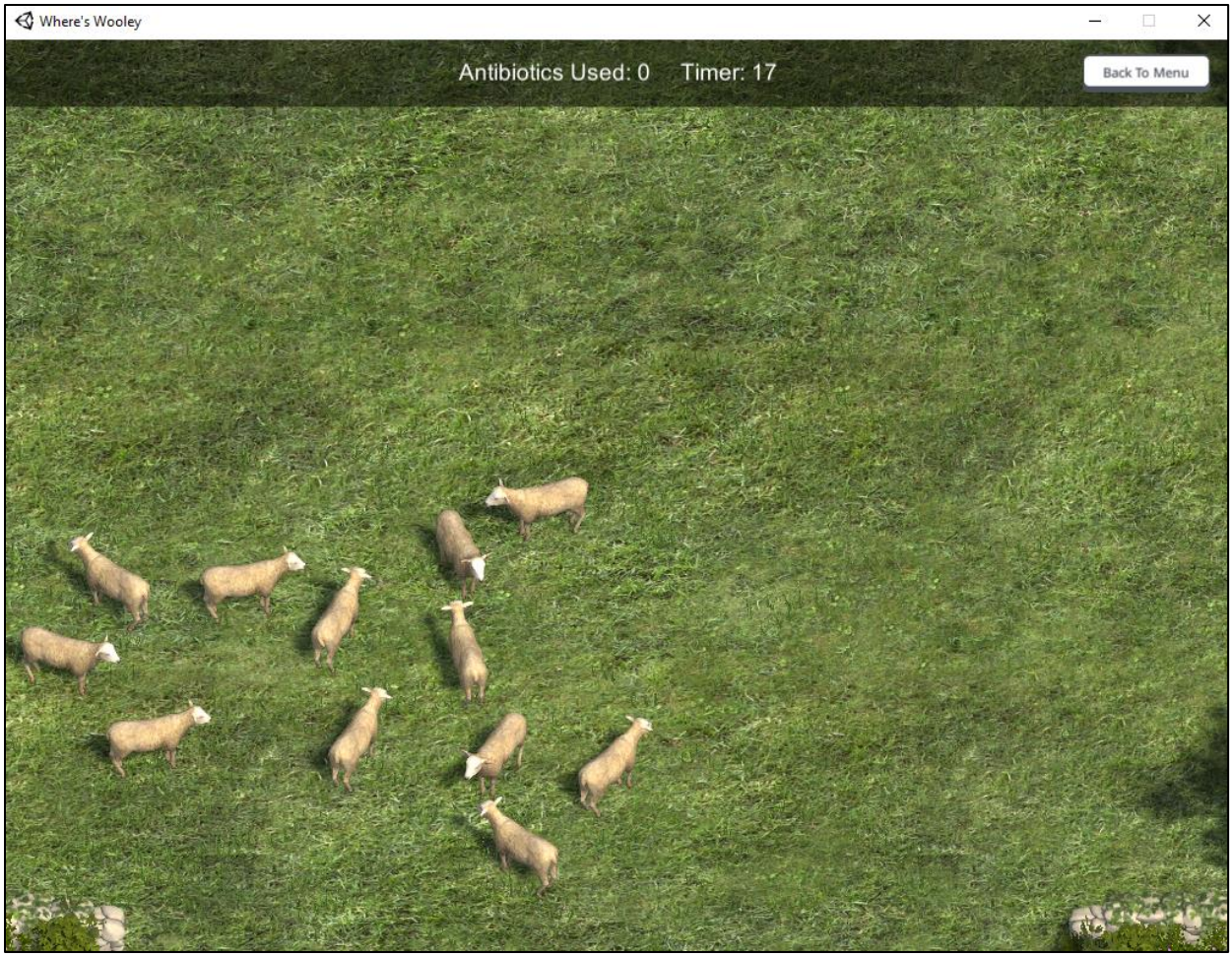


Figure 13: Screen – Active game

Figure 13 – Features (top to bottom):

- ***Antibiotics Used:** This counter counts the previous levels' antibiotics used plus the ones used on the current level.
- ***Timer:** The timer starts at 20 and shows how much time is remaining for this level. When it hits 0 the player will lose the game.
- **Back to menu:** This button allows the user to quit back to main menu anytime during the game.

*These are not available on the individual cases' Game screens to remove pressure and allow the user to focus on learning or practicing.

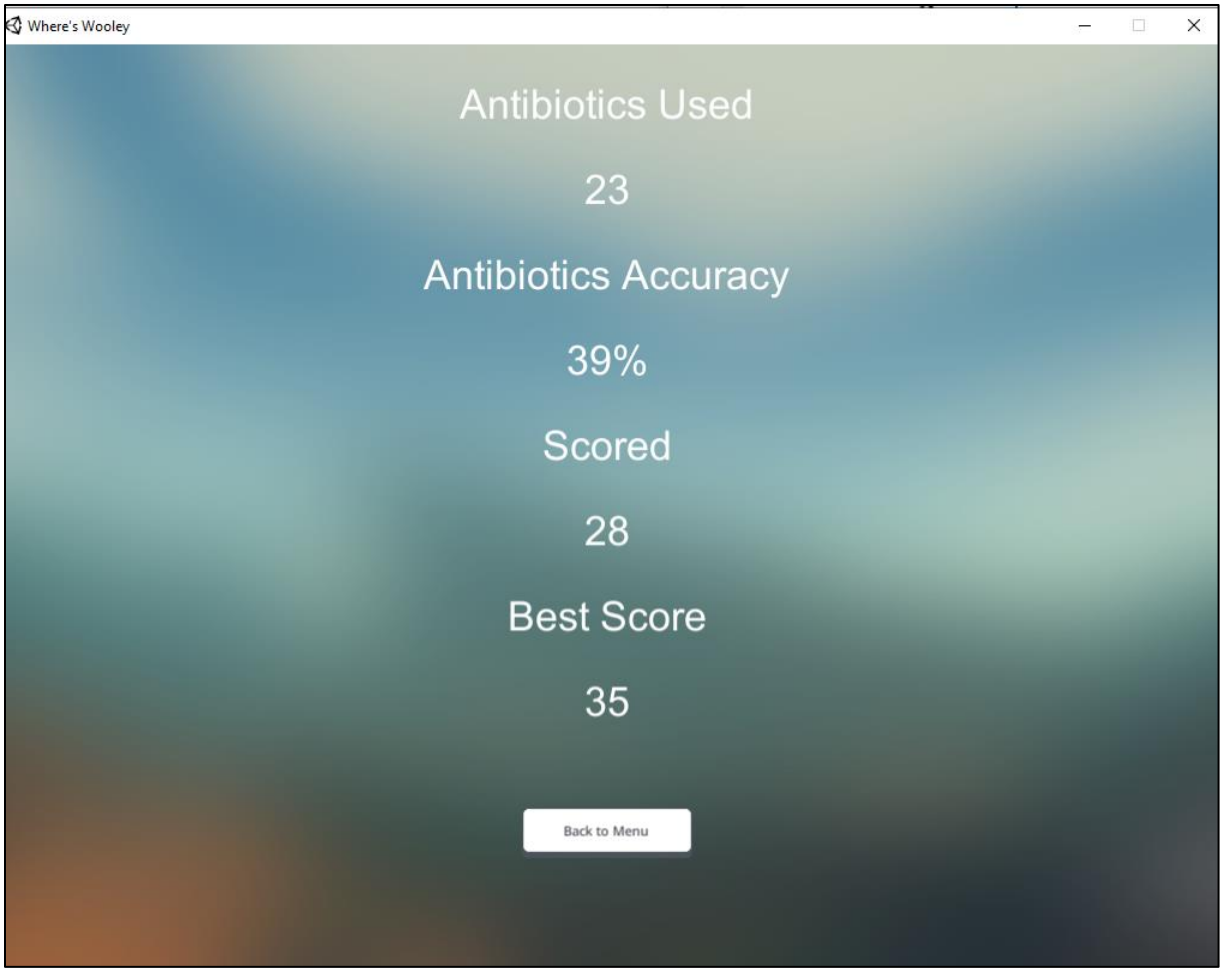


Figure 14: Screen - Feedback

Figure 14 – Features (top to bottom):

- **Antibiotics Used:** This tells the player how many antibiotics they had used in the game mode.
- **Antibiotics Accuracy:** The player loses accuracy for taking too long or mistreating a healthy animal.
- **Scored:** This shows the player their score based on accuracy.
- **Best Score:**
- **Back to Menu:** This button allows the user to quit back to main menu after the game and it also replaces the best score is current score is better.

4. Implementation

As a 3D editor that champions modularity, Unity is an efficient choice for the game. Each scene of the Unity game, there is a collection of objects, and each object, a collection of components and scripts with specialized responsibilities. Script based programming allows me to interact with the logic of any object readily. The Unity editor uses *Mono* which is compiled and ready for interpretation for quick feedback, reducing both the workload and durations of tasks. The inspector allows for the manipulation of script variables without the need to re-accessing code constantly and saves them respective to their scenes, making debugging and optimizing parameters quick. This is especially helpful knowing that Unity can interpret individual scenes and their scripts directly. The implementation of my scripts has modularity in mind, where applicable. Reusing scripts and components based on their environment is a practice that is considered throughout development.

Monobehavior is the base class of Unity scripts, a library of standard and helpful functions. It allows us to run scripts on load of an object with *Awake()* or framewise updates of in-game parameters as in *Update()* without needing to reinvent them. Unity has a long history of resources and documentation mostly written in C#, which is why this game's coding has been done in C#. I had used these as reference to gather the knowledge required to overcome implementation tasks. This allowed a smoother development process and the project to be reassured on implementing on more complex features.

Unity Engine also contains useful API I had used, such as:

- Data handling between system launches through *PlayerPrefs* class,
- An *Animation* controller class,
- Scene manipulation with *Scene Managers* class,
- And User *Input* handlers.

The points of discussion have amounted to Unity best being described as a *composition over inheritance* system (26). Figure 15 describes a summary of a generic scene of the game regarding its breakdown of entity-component relationship.

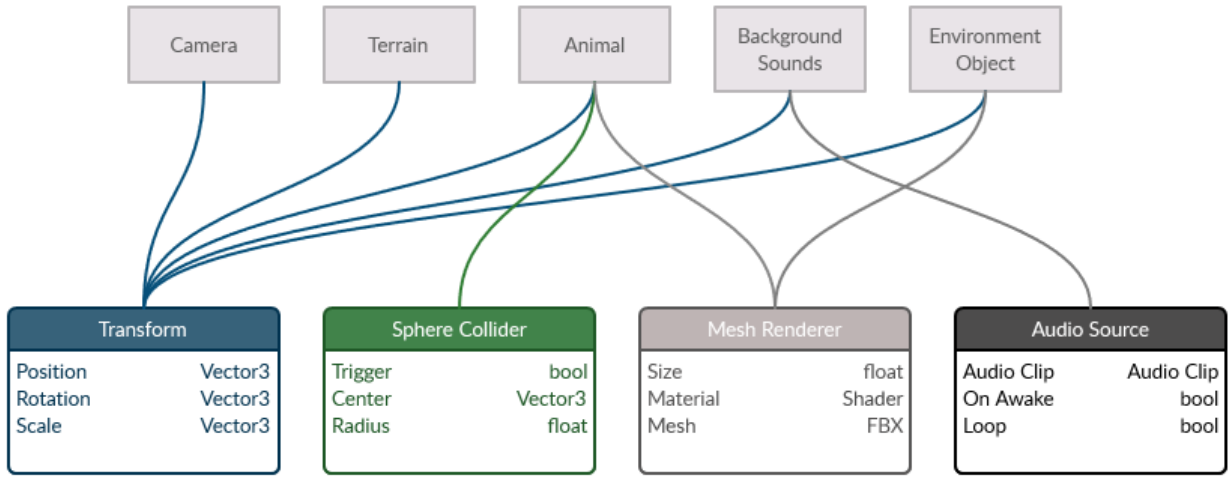


Figure 15: Example of component-entity relationship for the game

Unity handles player activity and input separate from the simulations but they can be specified to interact based on user requirements. Figure 16 summarizes the parts of the architecture of the system. The term handler refers to any type of script or object within the Unity logic that allows for the preceding functionalities. Due to Unity's component modularity, the scripts are broken down finely. Users can interact with the input handler, accompanied by simulation this brings about a change in the game state which is typically returned to the user. This allows for each functionality to be individually implemented and tested

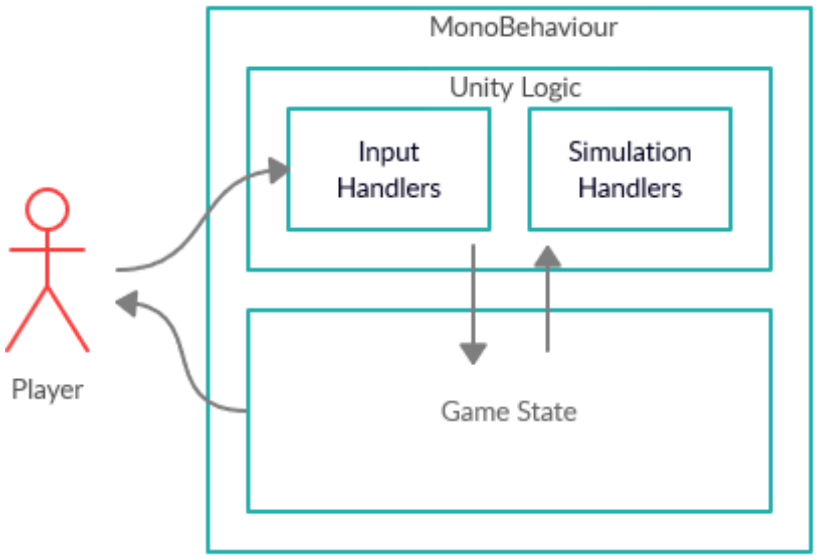


Figure 16: Software architecture breakdown of Unity

This section follows with the core building blocks of our game and how they have been implemented constructed to build a fully functional game. These sections include:

1. Wireframe development with JustinMind
2. Model and scene suitability with Unity Assets
3. Instantiating onto the scene through scripting spawners
4. Visual communication with animators and controllers
5. The flocking behaviour's implementation
6. Inputs, and data handling with the Game Data script
7. Sound implementation

a. The Wireframe

Before I had done Unity alongside User Centric Design, the team started with a wireframe to envision the mechanics of the game. Our team's original ideas are summarized well with Figure 17. I had used the prototyping software called *JustinMind* (27). JustinMind was easy to share and implement and update mechanics to play around with to see their feasibility and difficulty. The prototyping software benefits from having an export to HTML option which made it easy to share and run the files as most operating systems can access HTML natively. An example of this was when we had ideas about mass medication as a feature of the game. By a quick implementation, we quickly found out it was not a good idea because it defeats the premise of medicating animals individually through spotting signs of disease.

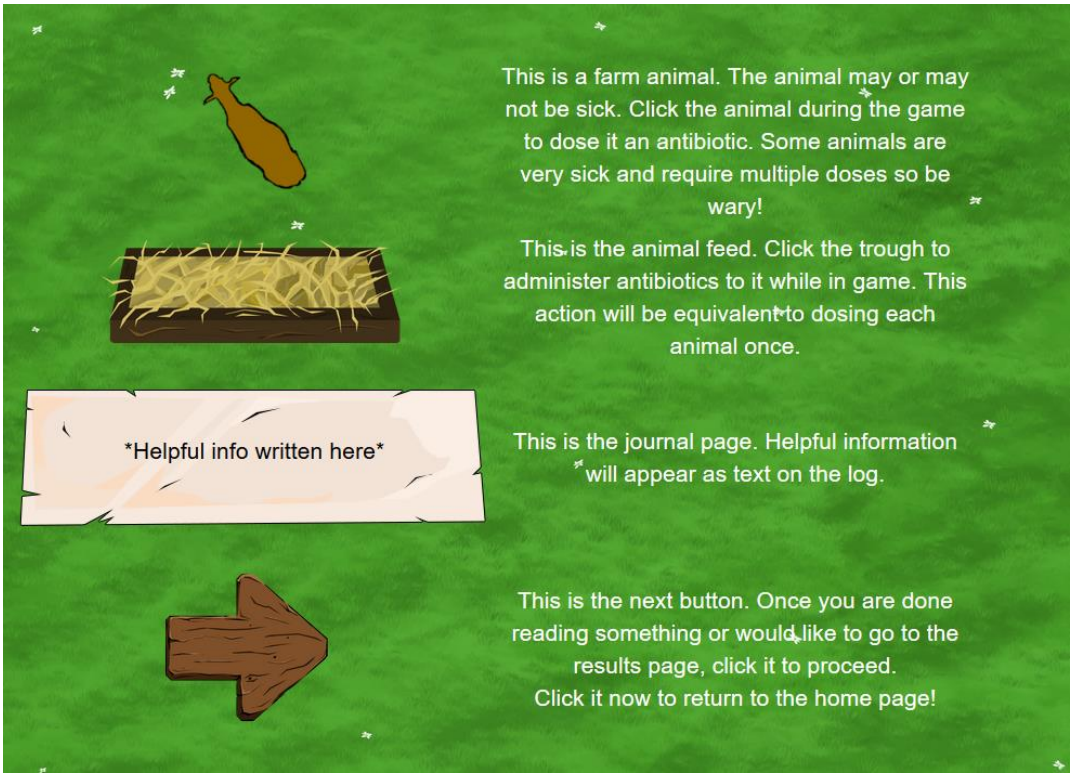


Figure 17: Wireframe's 'Guide' page

My only issues with the wireframing stage was that it was possible to start using Unity from the beginning to save time, especially since we had known that the game would benefit from being 3d environments over 2d.

b. Assets & Scene Building

Unity's asset store is a marketplace and community sharing hub of assets. Asset is a blanket term, describing any type of objects that can be used as part of the game project. This is useful as some of the complex models and assets are out of reach otherwise. The player would be playing the game on a 3d farm scene ran by the Unity engine. Taking realism into consideration, I needed to carefully select the type of assets the game would use. Mainly, the project needs realistic animal FBX models with emphasis on containing many detailed animations as well as high resolution textures. The original set of assets used in a previous prototype were too unrealistic for the farmers liking and had to be replaced because they did not meet their requirements. I had also learned that making it look like a farm would help

The main assets I had acquired and imported to use in the project include grassy terrain and detail textures, bushes, trees, drystone walls, and a lamb and sheep model. It was not wisest to continue without a second opinion so I had verified with the team to be sure our models and game objects met satisfactory realism. I also considered adding farm machinery but ultimately decided against it as the availible ones were not of an equal standard. For this project, this set of assets would suffice. I had also acquired some simple mobile style UI for flexibility and perhaps a future smartphone release. I also acquired background nature sounds and sheep sound effects for the audio requirements.

Figure 18: Assets & scene showcase

To save some time focusing on functionality, this projects has a minimally intrustive background scene. This included using the assets from the image in the corners of the screen as they don't visually interfere with the rest of the scene. Overall this resulted in a pleasant minimalistic yet realistic look, as in Figure 18, the team was satisfied with. Despite the high aesthetic look, I selected the lowest poly version to make sure textures load quicker and it runs on more graphical processing units.

The changes I had made to make these assets to make them suitable for our project were the following:

1. Scale their sizes to build a life scaled scene.
2. Remove all box colliders of all imported objects.
3. Add spherical colliders to only lamb and sheep prefabs.
4. Tagged the animals, where appropriate, as "Obstacle".

c. Spawning

To have our scene animals randomize positions, we use scripts which spawn our game animals. Spawning in Unity means to instantiate an asset onto the scene. The instantiated object can have customizable parameters which has been useful to tweak to optimize or change and add additional functionality such as animal types, positions, numbers etc. The game must have the player search for the sick animal, so the positions of the animals must shuffle per level. This functionality is especially important to the distancing case behaviour because the clue is relative position instantiated to the herd.

```
void Start()
{
    //Initial Sick animal instantiation
    var SickPos = new Vector3(Random.Range(-5f, 5f), 0f,  Random.Range(-5f, 5f));
    Vector3 center = SickPos;
    var InstSick = Instantiate(
        Animals[Random.Range(0, Animals.Length)], //Object to Spawn
        SickPos, //Position of Spawn
        Quaternion.Euler(new Vector3(0, Random.Range(0, 360), 0)) //Rotation of Spawn
    );
    InstSick.name = "Agent0";
    InstSick.transform.SetParent(Group.transform);
```

Figure 19: Sick animal spawner

A good example of importance of their arrangement is the case of distance. As in Figure 19, the start of our game level, the *Start()* function instantiates our sick animal, *Agent0*, on some reasonable position parameters facing any direction and within view of the player. Using Agent0's position as the center, the script generates an arc. Using parameters of the arc, the code Figure 20 returns random positions near the arc in Cartesian coordinates. These positions are checked to see if they coincide with any "Obstacle's" spherical collider. If they do, they are destroyed, and the function runs again up to three times per animal. This is to ensure the animals do not spawn in the same spot nor should the function attempt to find valid positions without infinitely looping.

```
Vector3 RandomCircle (Vector3 center, float radius, float start, float gap)
{
    float ang = start + Random.value * gap;
    Vector3 pos;
    pos.x = center.x + radius * Mathf.Sin(ang * Mathf.Deg2Rad) * Random.Range(1f, noise);
    pos.y = center.y ;
    pos.z = center.z + radius * Mathf.Cos(ang * Mathf.Deg2Rad) * Random.Range(1f, noise);
    return pos;
}
```

Figure 20: Arc position randomizer function

We can see the result of the script on Figure 22 and the results of location calculations.

Figure 22: Spawning - Distancing case

Figure 21: Spawning – Not-eating case

Unlike Figure 22, spawning for the not-eating case is required to be more random. The position is no longer a visual cue that the animal is ill. I had used the same previous script and changed only the parameters around the arc to have different spawning behaviour appropriate for the case. We can see the results on the Figure 21. The sick animal becomes indistinguishable from the rest of the herd by a distance metric. This is an example of how using scripts to spawn objects onto the scene is a useful method.

i. Spherical Colliders

Another useful adjustment that was made to improve the user interaction with the game was to increase the size of the spherical collider of the sick animal during its instantiation. This was done by checking if the name upon instantiation is Agent0 and then multiplying it by a number greater than one. This allows the system to give the users input the benefit of the doubt if they select an animal near the ill one as seen in Figure 23.

Figure 23: Spherical collider sizes

d. Animators

The animations of the project were included in the asset acquisition to bring about an important part of gameplay. Especially on cases such as the not-eating case, the behaviour is represented by only animation controls. On all levels it was important to use and script realistic behaviours, so the animals look authentic and natural as per requirements. Attached to the animal prefab is a script which calculates their states through level, environment interactions or time, but it is helpful to know any aspect could be parameterized. This makes any sort of behaviour based on any changes in the environment achievable.

The animal object is linked to a single set of animations through an animations controller. This is the component of an object that decides what the animal is doing via animation. With Unity, we are able to use one script and one controller to control all states of animation of one object, a good example of modular programming. The behaviour of the ill animal should differ from the rest of the herd in some cases. The script in Figure 24 identifies the name of the object it is attached to and what case level the scene is on and sets the animation parameters as in Figure 25.

```
anim.SetInteger("Level", PlayerPrefs.GetInt("level")%3);

if (gameObject.name == "Agent0")
{
    anim.SetBool("Sick", true);
}
```

Figure 24: Level based animator scripting

The determined parameters are then set on the animator. This animation tree links the transitions in animation made by the animals based on their parameters. A good example would be the not-eating case's animation tree as seen in Figure 25.

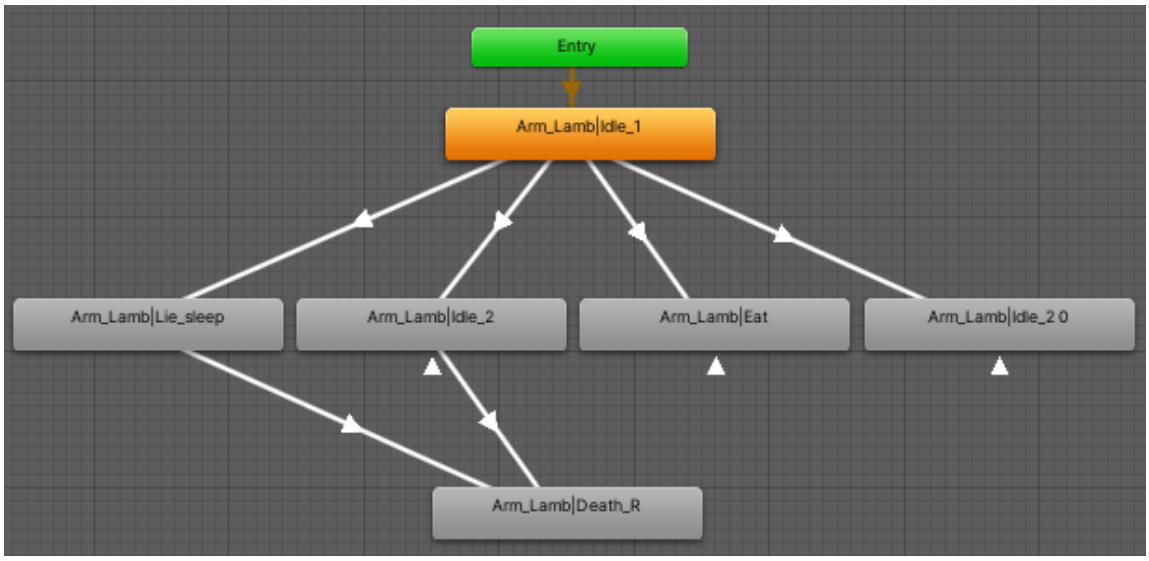


Figure 25: Animator's transition tree for not-eating & distance case of lambs

The player starts the scene with a few animals behaving idly and similar to one another. During this time, the animation controller has entered and is running the approximately 6 second animation "Idle_1". Once

this animation is done, the farm animals go into one of two states: “Idle_2” or “Eat” based on their set parameter from the previous statement. The player would then see a group of animals eating and also revealing one that is not, which is Agent0. This cycle repeats until the sick animal “dies”.

There were three problems that arose:

1. The healthy animals’ animations were in sync.
2. Death animation only came after the other animations were complete.
3. Flocking, or running, case had different parameter requirements for its animations.

To fix the sync problem, I first look into offsetting the animation. This proved to make the levels exponentially more difficult as it becomes a process of long and tedious elimination. The issue was that this was a indeed a realistic approach but I did not want players to struggle and become frustrated from the difficulty of the level. I turned to focusing on tweaking the animation speed of the animals, as in Figure 26, thinking I could get them to desynchronize over time instead so their “Eat” animation would be closer together. This method worked better than I had planned because the subtle details of the animals’ “Idle_1” and “Idle_2” proved more naturally differing speeds. The team agreed it looked less mechanical and more natural as a result.

```
void Start()
{
    anim = GetComponent<Animator>();
    anim.speed *= Random.Range(1f, 1.5f);
```

Figure 26: Animation speed desync

The second issue however, was easier to fix because Unity provides an option to break wait time for currently playing animation. Throughout this project, the death of an animal to occur instantly when conditions are met. Therefore for each transition to death, the exit times are removed.

Due to the slower pace of the sick animal, I had animated Agent0 to stay with the “Trot” animation instead of the “Run”. This made the sick animal more clearer as it looked to have difficulties moving around along with its slower pace.

The exception here was the flocking behaviour due to the sophisticated nature of the algorithm. The dependent factor of whether an animal was moving or not was purely based on whether it had animals in its surrounding, independent of its sickness. From the flocking algorithm, a separated animal stops moving. The problem was that the animal would appear to run in place when it did get separated. For that reason, a separate script was used to capture its parameters. I had to mend this through an adding an additional script to put the animal into its “Idle_1” state if there were no nearby neighbouring animals and as result.

e. The Flocking Behaviour

The gameplay relies on visual communication of behaviour to the player through animal animation and location. I had discussed the design of three commonly mentioned behaviours previously, but I emphasize that the game benefits from researching and implementing additional interesting behaviour cases to expand the library. The more cases the system has implemented, the more useful it becomes provided the cases are reasonably designed. Through demonstration and evaluation, I aim to prove that argument. It was therefore important that the current set of cases were well implemented. A good example of inventiveness of levels was the running case.

The flocking behaviour, for the running case, was the most difficult and interesting thing to implement. The flocking behaviour is an improvisational movement and direction based behaviour of an individual object and its surroundings (28). The intent is to replicate generic herd movements seen in groups of animals moving together, namely sheep. There are three non-trivial components to replicating the behaviour computationally:

1. *Cohesion*, an attraction between objects to form groups. This is calculated by finding all groups of objects local to each other and using their average position as a point of attraction.
2. *Separation* is repulsion between an object and objects around it. An object moves itself away from neighbouring objects in its specified vicinity by calculating their distances and inversely weighing their distance to calculate direction of the object.
3. *Alignment* is a group of objects rotating to face a common direction. This calculation is done by averaging out a local object's direction with its specified neighbours.

N.B. *Stay in Radius* is a property that keeps animals in range of the camera and users view and *ObstacleAvoidance* was used to test adding obstacles to the set for the animals to avoid.

Due to the complexity of this script, this project implements a modified version of a Unity algorithm by Board to Bits (29). This algorithm applied flocking behaviour to 2D objects on a plane, but this has been repurposed to suit this project's 3D environment by simply changing all Vector2 objects and calculations to Vector3 and using the Z axis in place of the Y axis. Because I was still operating on a 2d terrain despite being in a 3d environment, the script should not assign any values to the Y axis as to avoid Y directional flock behaviour.

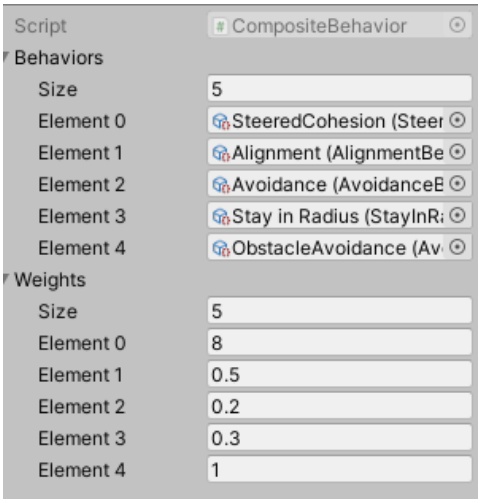


Figure 27: Weights of behaviours in flocking

This locomotion worked apart from one recurring visual bug where a few sheep would flicker due to the conflict between their alignment and separation. It was noticeable and distracting. The fix in Figure 27 was only partial; weights of separation and alignment were decreased so the flickering stopped but since the flock prioritized cohesion, this led to animals sometimes moving through one another instead. This is less visually distracting overall but use of this code needs to be refined or replaced. Despite its imperfections, I was thankful we were able to represent this scene by the end because it adds a difference of a moving object tracking challenge.

The greatest challenge faced in adapting the animations for this code was the animal would continue to move as a flock despite being in a dead state and animation. To find this solution was difficult, due to the complexity of the code but the implementation of a fix was rather simple. We find our time component and use it to determine the "speed" of Agent0. When the timer reaches zero, we set the speed to zero by multiplying the vector by 0 to stop the individual object resulting in Figure 28. With that the animal would stop changing its position and play the death animation for a visually consistent experience as in Figure 29.

```
if (move.sqrMagnitude > squareMaxSpeed){
    if (agent.name == "Agent0" && timer < 0)
    {
        move = move.normalized * maxSpeed * 0;
    }
    else if (agent.name == "Agent0")
    {
        move = move.normalized * maxSpeed * 3/5;
    }
    else
        move = move.normalized * maxSpeed;
```

Figure 28: Time and object dependent speed

Figure 28 shows our ill animal's max speed also cut by a fraction. This is to simulate the ill animal in the flock and the algorithm adapts to this change without inconsistency. This level's variation also kept things appropriately challenging. The process overall was a demonstrating experience that an interesting and unique variety of behaviours can and should be implemented.

Figure 29: Flock level at timer < 0

f. Game Data & Variable Handling

The Game Data script runs most of the variable handling and user interactions per level scene. These include antibiotic calculation and accumulation, the in-game timers, user's click functionalities on different game animals, and animation and scene management. This means it will be entirely reusable per scene with our Freeplay game mode.

When the Game Data object is initialized, it is initialized with the script of the same name. The Start functions assign a value for timer, the float 20 representing 20 seconds, which is then run through the Update function per frame to update the timer accordingly. This value had come out of independent testing sessions and with the advice of the expert. The purpose of the timer is more branched in functionality than explained in design.

```
if (timer < 0f)
{
    foreach (Transform child in Animals.transform)
    {
        if (child.name == "Agent0")
        {
            child.gameObject.GetComponent<Animator>().SetBool("Dead", true);
        }
    }

    dTimer -= Time.deltaTime;

    if (dTimer < 0)
    {
        SceneManager.LoadScene(0);
    }
}
```

Figure 30: Game Data timer < 0

If the timer runs out, i.e. reaches below 0, the player is set to lose the game. This should not happen immediately however, the player must be notified that the timer had run out. In Figure 30, the Update function changes the bool parameter of *Dead* to true of only the sick animal if timer hits below 0. Our animator controller then performs the Death animation. The game also returns the player to the main menu, Scene 0, but this should be done after the player sees the animation. When the timer variable reaches 0, changes in time are made to a secondary timer called *dTimer*. This timer is used as the means of delay between the player losing the game, seeing the animation and then the scene transitioning back to the main menu.

```
if (Input.GetMouseButtonDown(0) && timer > 0)
{
    Clicker(timer);
}
```

Figure 31: User input only during timer > 0

Figure 31 locks the user from any input during the delay timer described above to keep the player from progressing to another scene.

```
1 reference
void Clicker(float clock)
{
    RaycastHit hit;
    Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
    if (Physics.Raycast(ray, out hit, 100f))
    {
        if (hit.transform != null)
        {
            SphereCollider col;
            if (col = hit.transform.GetComponent<SphereCollider>())
            {
                if (hit.transform.gameObject.name == "Agent0")
                {
                    score = score + (25-(int)clock)/5;

                    if (SceneManager.GetActiveScene().name != "Flock")
                    {
                        PlayerPrefs.SetInt("level", PlayerPrefs.GetInt("level") + 1);
                        NextScene(PlayerPrefs.GetInt("level"));
                    }
                }
                else if (hit.transform.parent.name == "Animals")
                {
                    score++;
                }
```

Figure 32: The Clicker function; input handler

The *Clicker()* function, as in Figure 32, is the main function that processes user input and handles all data and scene changes with respect to the timer. This is helpfully modular code, making it easy to edit and update any and all changes in game state and variables regarding user's input. The Clicker is called by the previous if statement, user input is a click and timer is greater than 0, and projects a ray into the scene and detects the first *SphereCollider*, which is a collision box property to each prefab animal. If the ray detected is our sick animal's box collider, by checking for name "Agent0", we add the value of score with the above timer variate mathematical function and set next Unity scene through *NextScene*. Score the name of an integer variable that tracks total antibiotic usage. If the detected collider belonged to another animal however, the score variable is incremented by 1 only and no scene changes are made.

```
private void OnDestroy()
{
    PlayerPrefs.SetFloat("score", score);
}
```

Figure 33: Score saving across scenes

One thing to note about Unity is that no game objects are preserved across scenes. The score must be preserved, and this is done through Unity Engine's PlayerPrefs class. This class allows us to store and access preferences, or in this case variables, between sessions. In Figure 33, we use the *OnDestroy()* function to save score with PlayerPrefs between scenes and Start function to re-access them unless it is the first scene. We save the player's best score by a similar means to display on the menu page. It was also later learned that PlayerPrefs may cause Window's Defender to be suspicious of registry changes which needed to be explained often.

g. Sound Implementation

To lessen dullness of playing the game as well as keep the player immersed, the game needed to have audio background sounds and sound effects. Each appropriate level scene had to initialize background sounds. The game also should have contained sound effects of the sheep for realism and player immersion. I figured the best way to include click feedback and sound effects was to get the sheep to play their Baa sound as response. Simply, we add a game object called “Track” to all scenes. This prefab contains the sounds of wind and flora ruffling that was played on the Awake function on every loaded scene with Track. The Track object also contained the animal sounds as children objects with their Awake functions disabled. The script acted on an animal being clicked to play a random sheep sound.

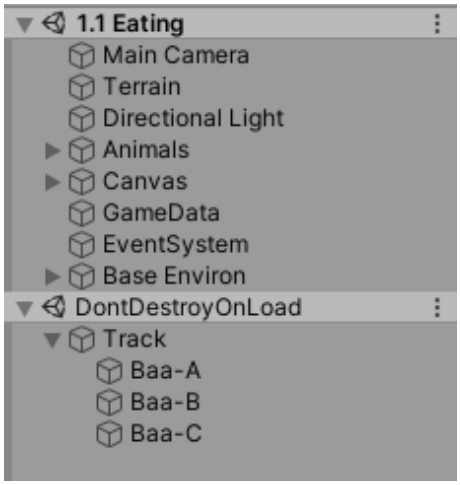


Figure 34: Sound objects moving between scenes

With Unity’s objects being disposed between scenes, an unexpected problem arose where the background sounds would start from the beginning at every level and the Baa sound effects were being cut off during scene transition. This is remedied by the function *DontDestroyOnLoad()* where we use it on the game object of the script to keep it from being removed between level transitions. Figure 34 shows the protected objects between scene transitions. This works well to protect the child objects from being destroyed during transition as well. With the addition of allowing the game object to be destroyed in the menu, it resulted in a smooth and immersive audio experience verified by the team.

5. Results & Evaluation

In this section I revisit the intended design and requirements and check if the project had met them as well as draw conclusions from the playtesting feedback questionnaires.

a. Test Cases

To ensure our core designs were implemented, I wrote out the test cases to verify that the use cases were met.

Case Name	Playing Freeplay		
Case Description	The player can use the software to play the main format of the game and receive feedback by the end of it.		
Pre Conditions	- Windows 7 SP1 or any later version - Graphics card with DirectX 10 or any later version - The program file has been downloaded		
Post Conditions	- The player has played freeplay mode		
Step No.	**Process**	**Response**	**Pass/Fail**
1	The game is executed	Game is running	Pass
2	The player clicks Freeplay from the menu	Player taken to game levels	Pass
3i	The player finishes the game levels	Player taken to feedback screen	Pass
3ii	The player clicks Back to Menu from the game levels	Player taken to main menu	Pass
3iii	The player loses the game levels	Player taken to main menu	Pass
4	The player clicks Back to Menu from the feedback screen	Player taken to main menu	Pass

Table 10: Test Case – Playing Freeplay

Case Name	Playing Cases		
Case Description	The player can navigate the software to the Cases menu and select individual cases. The player can then select a case and play it individually.		
Pre Conditions	- Windows 7 SP1 or any later version - Graphics card with DirectX 10 or any later version - The program file has been downloaded		
Post Conditions	- The player has played a case		
Step No.	**Process**	**Response**	**Pass/Fail**
1	The game is executed	Game is on main menu	Pass
2	The player clicks Cases from the main menu	Player taken to game levels	Pass
3i	The player clicks “Apart from the Herd”	Player taken to the respective scene	Pass
3ii	The player clicks “A Case of Lameness”	Player taken to the respective scene	Pass
3iii	The player clicks “Not Eating”	Player taken to the respective scene	Pass
4	The player clicks Back to Menu	Player taken to main menu	Pass

Table 9: Test Case – Playing Cases

Case Name	How to Play Instructions		
Case Description	The player can view the rules of the freeplay mode.		
Pre Conditions	- Windows 7 SP1 or any later version - Graphics card with DirectX 10 or any later version - The program file has been downloaded		
Post Conditions	- The player views/understands the rules of the game		
Step No.	**Process**	**Response**	**Pass/Fail**
1	The game is executed	Game is on main menu	Pass
2	The player selects “How to Play”	Player taken to Instructions scene	Pass

Table 11: Test Case – How to Play

Case Name	Change Settings		
Case Description	The player can access some settings and change them.		
Pre Conditions	- Windows 7 SP1 or any later version - Graphics card with DirectX 10 or any later version - The program file has been downloaded		
Post Conditions	- The player had set preferred settings		
Step No.	**Process**	**Response**	**Pass/Fail**
1	The game is executed	Game is on main menu	Pass
2i	The player clicks the gear icon from main menu	Player taken to settings	Pass
2ii	The player clicks the gear icon from case menu	Player taken to settings	Pass
3i	The player moves the volume slider	System does not save changes to the volume	Fail
3ii	The player selects a new resolution	System changes the resolution	Pass
3iii	The player clicks toggles full screen	System switches screen mode	Pass
3iv	The player resets their best score	System sets the PlayerPref best to 0	Pass
Comments: For 3iii, the full screen is sometimes incorrectly highlighted during windowed mode			

Table 12: Test Cases – Change Settings

The test cases yielded satisfactory results. The game is able to handles all important aspects of gameplay from the test cases Table 9 and Table 10. The instructions page also appears properly as in Table 11. All the important functionality from the use cases passed testing. The settings menu, Table 12, could use additional work as volume is broken and the other settings presets upon entering settings are loaded incorrectly. However, these are customizability features and not necessary to run the game as Unity adjusts to current screen size and resolution and volume can be adjusted from the operating system.

i. Team Testing

Apart from the test cases, I had team members test the game constantly to ensure it was working. Originally a few bugs were caught such as animals disappearing or the textures not loading correctly but I had quickly attended to them and sent them back for testing. I had no additional complaints on bugs appearing past that.

b. Playtest Questionnaire

To determine whether our game was successful and what purpose it serves, I had written up a feedback questionnaire to gather information. This questionnaire uses the MEEGA+ model questions (30), as similar games such as AntibioGame had used with good result. I had adjusted the questionnaire to be less about education and more about recommended purpose. I felt the best way to understand how a person would use our product is through what type of product he/she would recommend it as, if at all. The questionnaire, Appendix C, also uses a Likert scale to assist in analyzing the data. The questionnaire was sent to farmers, vets, and other health experts with the game resources to linked through file sharing. The results had been displayed on a 100% stacked bar as in Figure 35.

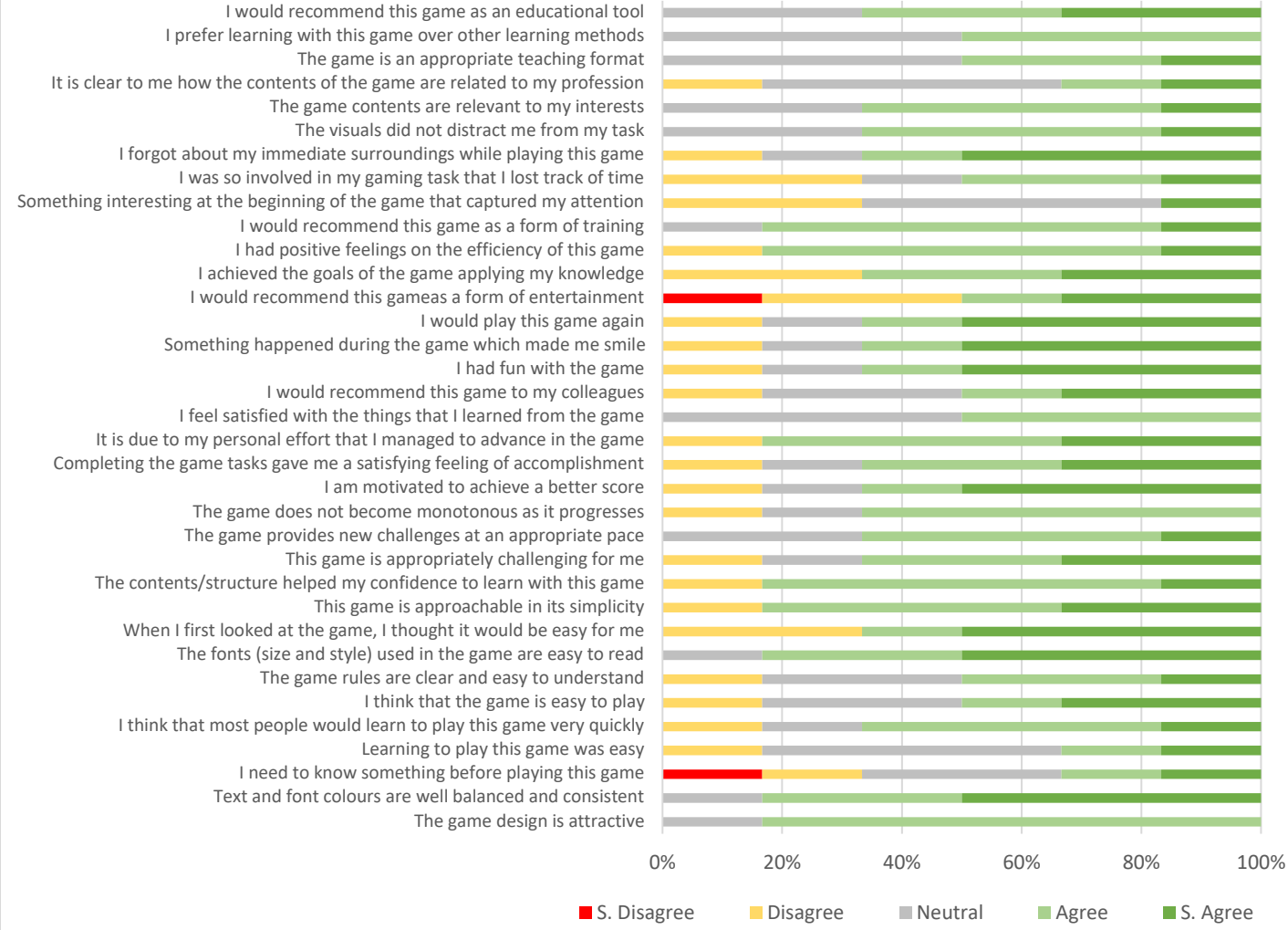


Figure 35: 100% stacked bar on questionnaire feedback

Evaluator	Section	Comments
A – Veterinary Microbiology Academic	Improvements	"…a textbox or introduction screen at the start of every gameplay could help with player immersion. I'm thinking of a dynamic and short paragraph explaining the role of the player (veterinarian, technician, farmer, etc) and the importance towards not only treating the disease but being aware of contributing to antibiotic resistance."
B – AMR Academic	Additions	"Clearer cause and effect – e.g. sheep dying, more animals showing symptoms, feedback on whether you treated correctly at the time of application so you learnt more as you progressed"
C – Healthcare AMR Academic	Comments	"It was strangely addictive – only stopped because had to get on with work!"
D – Health Sciences Academic	Additions	"Different camera angles. Cows."
E – Engineering Academic	Improvements	"More sheep, and more subtle signs. You learn to look out for certain behaviour."

Table 13: Feedback extracts table

i. Initial Feedback

The questionnaire had been sent to mostly farmers, but most of our participants that sent feedback were not farmers or vets. During the feedback collection stage, a few emails were received regarding difficulties with the file sharing and extracting from archives – one the participants listed it as a bug. I had realized the few necessary technical steps in accessing the game files may not have been considered enough. There was an effort to assist in the access by the creation of instructional documentation, but it did not seem enough. These user hurdles should be considered more in the future. Due to this accessibility issue, there is a lack of farmer feedback on this area of the evaluation. The evaluation relies on experts of other disciplines to evaluate the game under study.

ii. Discussion

Keeping that in mind, the feedback analysis shows there is interest through a majority positive response on every dimension of *Where's Woolly*, the name of this project's game. Figure 35 shows 50% of the participants strongly agreed they were drawn into the game, forgetting their immediate surroundings thanks to the gameplay loop and level design. One user also mentioned it was "strangely addictive". Moreover, there was a strong positive response on the dimension of Fun, including enjoyment and replayability. The visuals were well received thanks to the careful selection and use of each asset on the game scenes. No in-game bugs were reported either providing a smooth experience for the users. Overall, the game succeeds in creating an enjoyable and attentive experience for its players.

Where's Woolly did however lack an initial hook element, as only one participant from Figure 35 was intrigued upon first impression. Some more visually enticing features, perhaps on the menu or initial scenes would stimulate more curiousity. The users also provided us with additional directions for improvement as in Table 13. *Where's Woolly* should invest in more into explanations about the happenings in the game and some instruction about them. An explanation could be provided to the player via dialog or narrative about the events of the farm in the game. More work regarding the feedback loops and context of the challenges the player is overcoming; it needed to be clearer on the learning outcomes and perhaps even explicitly state facts about AMR on farms. Participants were also interested to see more of was the signs and symptoms. As mentioned before, the animation plays a strong role in the game and would benefit from additions in its behaviours such as different signs, multiple animals being ill, animals dying. Naturally more cases and behaviours of interest should be added to the game. These could include expanding the set of animals to cover a broader set of behaviours. Overall, the gameplay structure is effective and would benefit from more contextual information as well as increasing the library of available cases.

The most important part of the data, however, is regarding part of our aim in finding the potential purpose for *Where's Woolly*. In the recommendation questions, despite the Fun section rating high in the questionnaires, the game was not highly recommended as a means of fun. The most agreed upon purposes were training and education. By these two metrics, it is acknowledged by the users that *Where's Woolly* is a serious game which is remarkable as it had never been presented to them as a serious game. Therefore, *Where's Woolly* is recognized to teach and offer something more than just entertainment.

6. Future Work & Recommendations

Unity's compatibility with mobile technology would keep it accessible to farmers, as we had learned some of them had trouble with downloading and extracting our prototype files. Sharing should be more accessible in the future. Farmers in the interview also stated their interest in mobile technology as well as their usability for a product like this. From the interviews, we became aware that tablets are also a form of technology they typically use. This should be taken into consideration.

As the questionnaire feedback had made a point, the game needs more context communicated to the player. The "How to Play" screen was an attempt to add that, but it may be worth investigating having a narrated format of play, perhaps a tutorial or story mode. The communication can also be achieved by personalized feedback in the assessed play modes, with not only text feedback but game level changes i.e., more animals get sick the more antibiotics wasted per level.

Past that point, the next step for the game would be experimenting with deeper difficulty options. The simpler solution is to play with the number of animals but more creatively hybrid levels where multiple symptoms are available with also with varying degrees of severity via a more sophisticated animator tree. The game should make some new attempts to fool the player in thinking an animal is sick. This would be an interesting approach to difficulty, but I would need to do research on the signs that commonly do trick farmers, if any at all. The animations of the project's assets could be utilized in more ways to create additional behaviours as well, especially now that we have gathered that the game model works and intrigues the players. Increasing the library of levels is one again a highlighted improvement. It makes levels more varied and intriguing as well as offers new insights through learning and training material on what other behaviours are not obvious.

Building on these points, I suggest an artist proficient in modelling animals and animations be added to the project because then the behaviours can be directly modelled and animated as per farmer and vet description. The accuracy in their symptomatic behaviours will make the product more realistic and therefore usable as a training and educational tool. Moreover, some characteristics on the model can also be displayed well such as growing sickly and including signs such as bleeding. The project will benefit from increased flexibility of assets and can meet tighter and more specific requirements.

Apart from increasing the amount of behaviours, the idea of a variety of animals has been suggested by some farmers. If the behaviours of one animal have been modeled well enough, it would be natural to expand into understanding more animal behaviours. If there are any unique to certain animals, that would be an interesting place to begin. More animals naturally offer more behaviours, movements, symptoms, characteristics, and subtleties. Variety could also mean different treatment methods. Vaccines have been mentioned by the farmers multiple times, this may be worth designing and implementing unique behaviors and responses to each of these.

The main functional requirements that had been set through research had mostly been met apart from some trouble working with the settings page. Repairing some codes would be a good start and adding a few more personalization settings. The spawner script perhaps offers a little too many functional variable inputs and can be reduced to the most important ones. A better input feedback system and adding a few extra camera views to switch between would be optimal too. Apart from the minor technical improvements that can be made, the levels of the game work and project achieves its goals.

Another idea that came up twice in the interviews was seasons. The idea of different weather effects and seasonal conditions could be tested and implemented with good research. Improving the environments in the levels may prove useful to keep farmers intrigued but should not be too distracting as the assessment is based on time as well. This requires a deeper level design structure as well as the appropriate assets. Signs and red herrings can also be used to trick the farmers for increasing difficulty, but cautiously researched and implemented as the game should never provide an unintuitive solution. This is to get players to realize there is enough signs and symptoms to figure the game out and it is not left to chance. This would hopefully reflect that diagnosing sick animals is not a gamble in reality and that good practice can be achieved.

7. Conclusions

The main aim was to explore the implementation of antibiotic practices as a serious game in a way to which farmers can relate. We have discussed the serious games industry and their uses in training have evolved and to include training future doctors. As technology progresses, there becomes new ways to express old problems and even more ways to share them. I had also outlined a distinct lack of AMR in agriculture games. *Superbugs* (31), *AntibioGame* (19) and the others were all great sources of inspiration and helped me in my designs and requirement gathering. The main objective was to understand what makes serious games effective and innovative in their uses and implement them into *Where's Woolly*.

I had worked on an initial Unity prototype of a serious game regarding antibiotics practices in agriculture to present to farmers. During the initial round of interviews, it was found that the game was too unrealistic through both visuals and mechanics. Applying User Centered Design had impacted the direction of the game greatly and therefore it was perhaps the best approach to use for this game.

I believe *Where's Woolly* had captured most of those elements according to the positive feedback. The feedback was overwhelmingly positive. The testers said they see it primarily as an educational and training tool which aligns with the objectives of a serious game. The main gameplay mechanics were well received, and the game had no bugs.

There were points of helpful constructive criticism that were received. The game could be more communicating with the player via tutorials, instructions or narrative and the context of the problems could be better highlighted through these methods. The game would benefit from adding additional behaviours and cases as well as different types of animals. Unity software architecture allows this to be easily implemented.

8. Reflections & Learning

During the project, I had learned a few valuable lessons. One of the most important ones was the importance of firsthand feedback. I learned that even without writing code, you can get far into a project by communicating to understand it better. In the future, I will be even more willing to engage and have understood the value discussions bring to innovative and interesting ideas. I had learned more from my first interview than I had in the preceding week of programming. Identifying valuable sources of information is an important skill for finding the right direction in a project.

I learned a lot about the Unity editor too. I had never worked extensively with animators before and have improved my usage of them over the course of the project to the point where I was able to create modular animations with scripts. I had also never worked with the intention of simulating realism before, as my interests are on entertainment. Coming out of my comfort zone however taught me a lot about how real-world problems can be studied through a technical lens, especially one as large and threatening as antimicrobial resistance.

I have also learned to appreciate project management skills and their importance. Project management would allow me to pace tasks better and increase my productivity. My supervisor had provided me with many different methodologies such as User Centered Design which I had applied for the first time. Project design with a user-oriented method had taught me more on interpreting and understanding feedback. Thematic analysis was also quite an interesting way to extract thoughts and ideas from interviewees. I had taken a while to adjust reading between the lines of a transcript, but I am faster and more confident at the task now.

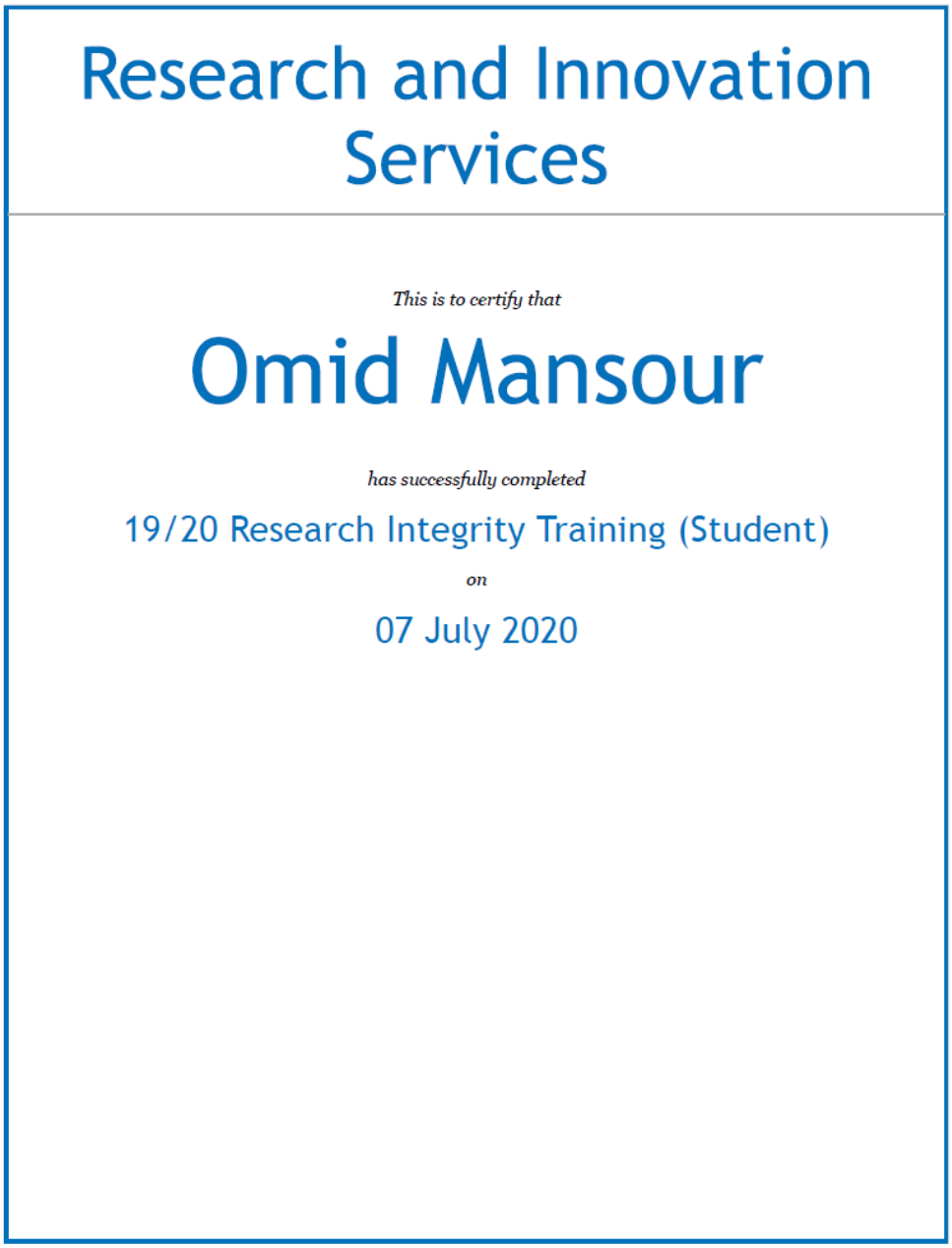
9. References

1. Alvarez J, Djaouti D. An introduction to Serious game Definitions and concepts. Serious Games & Simulation for Risks Management. 2011;11(1):11-5.
2. O'Neill J. Antimicrobials in agriculture and the environment: reducing unnecessary use and waste. The review on antimicrobial resistance. 2015:1-44.
3. Kyaw BM, Car LT, van Galen LS, van Agtmael MA, Costelloe CE, Ajuebor O, et al. Health professions digital education on antibiotic management: systematic review and meta-analysis by the digital health education collaboration. Journal of medical Internet research. 2019;21(9):e14984.
4. Bellotti F, Ott M, Arnab S, Berta R, de Freitas S, Kiili K, et al., editors. Designing serious games for education: from pedagogical principles to game mechanisms. Proceedings of the 5th European Conference on Games Based Learning University of Athens, Greece; 2011: Academic Publishing Limited.
5. Nisansala A, Weerasinghe M, Dias G, Sandaruwan D, Keppitiyagama C, Kodikara N, et al. Flight simulator for serious gaming. Information science and applications: Springer; 2015. p. 267-77.
6. Jumper E, Baughn J, editors. The use of Microsoft Flight Simulator in aerospace education. 9th Applied Aerodynamics Conference; 1991.
7. Formosa P, Ryan M, Staines D. Papers, Please and the systemic approach to engaging ethical expertise in videogames. Ethics and Information Technology. 2016;18(3):211-25.
8. Chalmers A, Debattista K, editors. Level of realism for serious games. 2009 Conference in Games and Virtual Worlds for Serious Applications; 2009: IEEE.
9. Dieleman H, Huisingh D. Games by which to learn and teach about sustainable development: exploring the relevance of games and experiential learning for sustainability. Journal of Cleaner Production. 2006;14(9-11):837-47.
10. Brown E, Cairns P, editors. A grounded investigation of game immersion. CHI'04 extended abstracts on Human factors in computing systems; 2004.
11. Ennis C, Hoyet L, Egges A, McDonnell R. Emotion capture: Emotionally expressive characters for games. Proceedings of Motion on Games2013. p. 53-60.
12. Wattanasoontorn V, Boada I, García R, Sbert M. Serious games for health. Entertainment Computing. 2013;4(4):231-47.
13. Encarnação M. On the future of serious games in science and industry. Proceedings of CGames. 2009:9-16.
14. Graafland M, Schraagen JMC, Schijven MP. Systematic review of validity of serious games for medical education and surgical skills training. The British journal of surgery. 2012;99(10):1322-30.
15. Farrell D, Kostkova P, Lazareck L, Weerasinghe D, Weinberg J, Lecky DM, et al. Developing e-Bug web games to teach microbiology. Journal of Antimicrobial Chemotherapy. 2011;66(suppl_5):v33-v8.
16. Farrell D, Kostkova P, Weinberg J, Lecky D, McNulty C. Online games teaching children hygiene and antibiotic resistance: Evaluation of the e-Bug games. International Journal of Infectious Diseases. 2010;14:e43.
17. Servitje L. Gaming the Apocalypse in the Time of Antibiotic Resistance. Osiris. 2019;34(1):316-37.
18. Resistance RoA. Tackling drug-resistant infections globally: final report and recommendations: Review on antimicrobial resistance; 2016.
19. Tsopra R, Courtine M, Sedki K, Eap D, Cabal M, Cohen S, et al. AntibioGame®: a serious game for teaching medical students about antibiotic use. International Journal of Medical Informatics. 2020;136:104074.
20. Sutherland L-A. The 'Desk-Chair Countryside': Affect, authenticity and the rural idyll in a farming computer game. Journal of Rural Studies. 2020;78:350-63.

21. Polygon. Farming Simulator's biggest fans are farmers, dev says 2013, Aug 26 [Available from: https://www.polygon.com/2013/8/26/4654126/real-farmers-love-farming-simulator-giants-software-gamescom-2013.
22. Abras C, Maloney-Krichmar D, Preece J. User-centered design. Bainbridge, W Encyclopedia of Human-Computer Interaction Thousand Oaks: Sage Publications. 2004;37(4):445-56.
23. Braun V, Clarke V. Thematic analysis. 2012.
24. Guardiola E, Czauderna A, editors. Merging gameplay and learning in educational game design: the gameplay loop methodology in antura and the letters. ECGBL 2018 12th European Conference on Game-Based Learning; Academic Conferences and Publishing Limited: England, UK; 2018.
25. Vesselinov R, Grego J. Duolingo effectiveness study. City University of New York, USA. 2012;28.
26. Knoernschild K. Java design: objects, UML, and process: Addison-Wesley Professional; 2002.
27. Farrell-Vinary P. Justinmind. ACM SIGSOFT Software Engineering Notes. 2011;36(3):34-5.
28. Reynolds CW, editor Steering behaviors for autonomous characters. Game developers conference; 1999: Citeseer.
29. Board to Bits. flocking-algorithm 2019, Mar 14 [Available from: https://github.com/boardtobits/flocking-algorithm/tree/master.
30. Petri G, von Wangenheim CG, Borgatto AF, Lee N. MEEGA+, Systematic Model to Evaluate Educational Games. 2019.
31. Prize L. Superbugs: The Game. 2018.

10. Appendices

i. Appendix A



Research and Innovation Services

This is to certify that

Omid Mansour

has successfully completed

19/20 Research Integrity Training (Student)

on

07 July 2020

ii. Appendix B

Study Protocol

Part 1: Interview Questions

The aim of this interview is to understand the management of disease in livestock production, especially in the context of antibiotic use and the pressure on farmers to reduce antibiotic use. As well as practical questions concerning the participant's background and farm operations, there are more open-ended questions where participants may more freely discuss their thought processes, feelings and opinions on the subjects. This interview will take up to 30 minutes.

Previous career experience

1. How long have you worked in your current industry (e.g. farming, veterinary)?
2. Roughly how many farms have you worked on?
3. Roughly how many animal herds have you worked with?
4. What is your current role and how long have you held it?

Current job experience

5. Approximately how many animals do you currently work with? Please also tell us what types (e.g. cattle, calves)?
6. What production system(s) do you work with (e.g. outdoor, indoor-bred)?
7. Does your farm or the farms you work with have any classifications (e.g. Organic, Red Tractor, Soil Association, RSPCA Assured)?

Managing animal health

8. In what ways do you monitor animal health (e.g. regular weight or temperature checks)?
9. If you spot signs of disease, how do you respond?

Antibiotic use in livestock production

10. What are the most common reasons antibiotics are used in the livestock production? Please give general and specific reasons.
11. How frequently are antibiotics used in livestock production (e.g. once a day, once a week, once a month)?
12. Why and how would you apply antibiotics to A) individual animals B) a group of animals?
13. What might cause a farmer to change their antibiotic use strategy?
14. There's pressure on farmers to reduce their antibiotic use. Do you agree with this? From your perspective, what might prevent farmers from reducing their antibiotic use? What would enable them to reduce their antibiotic use?
15. Are you aware of current legislation/have you got experience with policy (interventions) aimed at reducing ABU? If so, what are they and how do you feel about the approaches taken?

Part 2: Prototype Discussion

After the initial interview, a game prototype will be provided to the participants. They will be asked to go through the tutorial, a short story walking using the interface, by themselves and try three preset levels. This would take about 5 minutes. Unless asking for assistance, the participant will be allowed to freely explore the prototype to observe a natural user response. Once the demonstration is over, the participant will be asked:

- What is the first thing that comes to mind after that demonstration?

A rigid line of questioning may not bring up the most apparent or standout aspect of their experience. After that, some more straightforward questions to understand their ideas and preferences around technology, and video games.

General

1. Was it enjoyable?
2. Was the goal of the game clear?
3. Do you think it reflects reality – which elements are captured, what was missing?
4. Did you learn anything/gain any insights from playing the game?
5. How do you think the game could be used?

Design

6. Was the game difficult to navigate and if so, how could it be made easier?
7. Can you list what forms of technology you have access to daily or at home? Which one of those platforms would you prefer this on?
8. Did you like the theme? Did you like the art style? Did you like the buttons and interface?
9. Do you think viewing the farm overhead affect your decision making/practice? [*Show non overhead image*] Would you prefer/act differently if this was the point of view?
10. What comments do you have on the themes or design choices? Do you have any design or interface suggestions?

iii. Appendix C

User Testing

Goals

The goal of the user testing is to test our game prototype against experts and potential users. This would be through having the participants interact with our prototype and give us feedback through MEEGA+ Model questionnaire along with some open-ended questions. From this testing exercise, I hope to validate areas successful in representation and understand areas that need further work.

Questions

Feedback Questionnaire

We breakdown areas of our game and ask them how much they agree with our assessment on MEEGA appointed aspects on a Likert scale 1-5 (5 being Strongly Agree):

Dimension	Subdimension	Item No.	Description
Usability	Aesthetics	1	The game design is attractive
		2	Text and font and colours are well balanced and consistent
	Learnability	3	I need to know something before playing this game
		4	Learning to play this game was easy
		5	I think that most people would learn to play this game very quickly
	Operability	6	I think that the game is easy to play
		7	The game rules are clear and easy to understand
	Accessibility	8	The fonts (size and style) used in the game are easy to read
Confidence		9	When I first looked at the game, I had the impression that it would be easy for me
		10	This game is approachable in its simplicity
		11	The contents and structure helped me to become confident that I would learn with this game
Challenge		12	This game is appropriately challenging for me
		13	The game provides new challenges (offers new obstacles, situations, or variations) at an appropriate pace
		14	The game does not become monotonous as it progresses (repetitive or boring tasks)
		15	I am motivated to achieve a better score
Satisfaction		16	Completing the game tasks gave me a satisfying feeling of accomplishment
		17	It is due to my personal effort that I managed to advance in the game
		18	I feel satisfied with the things that I learned from the game
		19	I would recommend this game to my colleagues
Fun		20	I had fun with the game
		21	Something happened during the game (game elements, competition, etc.) which made me smile

		22	I would play this game again
		23	I would recommend this game as a form of entertainment
Competence		24	I achieved the goals of the game applying my knowledge
		25	I had positive feelings on the efficiency of this game
		26	I would recommend this game as a form of training
Focused attention		27	There was something interesting at the beginning of the game that captured my attention
		28	I was so involved in my gaming task that I lost track of time
		29	I forgot about my immediate surroundings while playing this game
		30	The visuals did not distract me from my task
Relevance		31	The game contents are relevant to my interests
		32	It is clear to me how the contents of the game are related to my profession
		33	The game is an appropriate teaching format
		34	I prefer learning with this game over other learning methods
		35	I would recommend this game as an educational tool

Order the cases (Feeding, Running, and Distancing) from least to most difficult.

Least Difficult	Moderate	Most difficult

What was your best score?

Did you come across bugs or any issues with the game's functionality?

What improvements would you suggest to the game?

If you could add something to the game, what would it be?

Additional Comments