Final Design Report

Team Infinity

Environmental Pet Simulator

Cara Fisher c086f573@ku.edu

Paul Dreger pdreger@ku.edu

Hans Brown h458b873@ku.edu - ***Contact***

Jackson Schilmoeller j427s666@ku.edu

Daniel Murga d060m098@ku.edu

Project Description

- Why is the project being undertaken?
    - We wanted to do a VR project. Some of us wanted to do virtual environments, and others wanted to make a game. Given that the scope of recreating the real world in virtual space is outside of current Vive and Oculus Rift capabilities, we decided on a VR pet game.
- Describe an opportunity or problem that the project is to address:
    - There has been substantial research showing that owning a pet improves health (see works cited). If one can not have a pet due to limiting physical capability, allergies, housing contracts, or any other reasons, then a real life pet is infeasible, and those health benefits are lost. Therefore, we decided to make a VR pet simulator to allow for those affected to hopefully gain some of the health benefits.
- What will be the end result of the project?
    - An interactive virtual pet living inside a virtual user-customizable model of a given living room.

Project Milestones

- First semester
    - Determine feasibility of room mapping
        - October 27, 2017
    - Simple pet model inhabits and moves around room
        - November 10, 2017
    - VR Hello World
        - October 10th, 2017
    - Pet Needs System in place
        - December 10th, 2017
- Second semester
    - Model animations finalized
        - March 26th, 2018
    - Pet interacts with surroundings based on some limited AI (neural network)
        - February 28th, 2018
    - Player movement through room
        - March 26th, 2018
    - Pet AI final desire weighting
        - March 26th, 2018
    - Player can model create a room
        - March 1st, 2018
    - More choices of pets
        - April 18th, 2018

** Gantt Chart in attached PDF document **

    - To view original Gantt chart in Gantter, download using link: https://drive.google.com/open?id=172NJpONGTKLJFFo5tGS5mg8xNoDaRxtN, then open using the Gantter website (requires free account) or the Gantter for Google Chrome App.

Project Budget

- Hardware, software, and/or computing resources
    - Unity Game Engine Personal Edition (free)
    - HTC Vive (provided through the school)
- Estimated cost
    - Unity Game Engine Personal Edition: Free, project is non-profit
    - HTC Vive: $599
- Vendor
    - Steam for HTC Vive
- Special training (e.g., VR)
    - Unity Game Engine (free)
    - Unity VR API
        - https://unity3d.com/learn/tutorials/topics/virtual-reality/vr-overview?playlist=22946
- When they will be required?
    - VR Headgear is needed as soon as we can begin testing.
        - Testing begins after we finish researching VR implementation in the Unity Engine

Work Plan

- Paul will handle implementation of Pet Logic, the Pet-Player interaction system, and Pet Artificial Intelligence
- Cara will handle VR implementation and the Game Model
- Jackson will handle VR implementation, UI, and Virtual World Building
- Hans will handle layout design, graphics, and the Game Model
- Danny will handle Virtual World Building

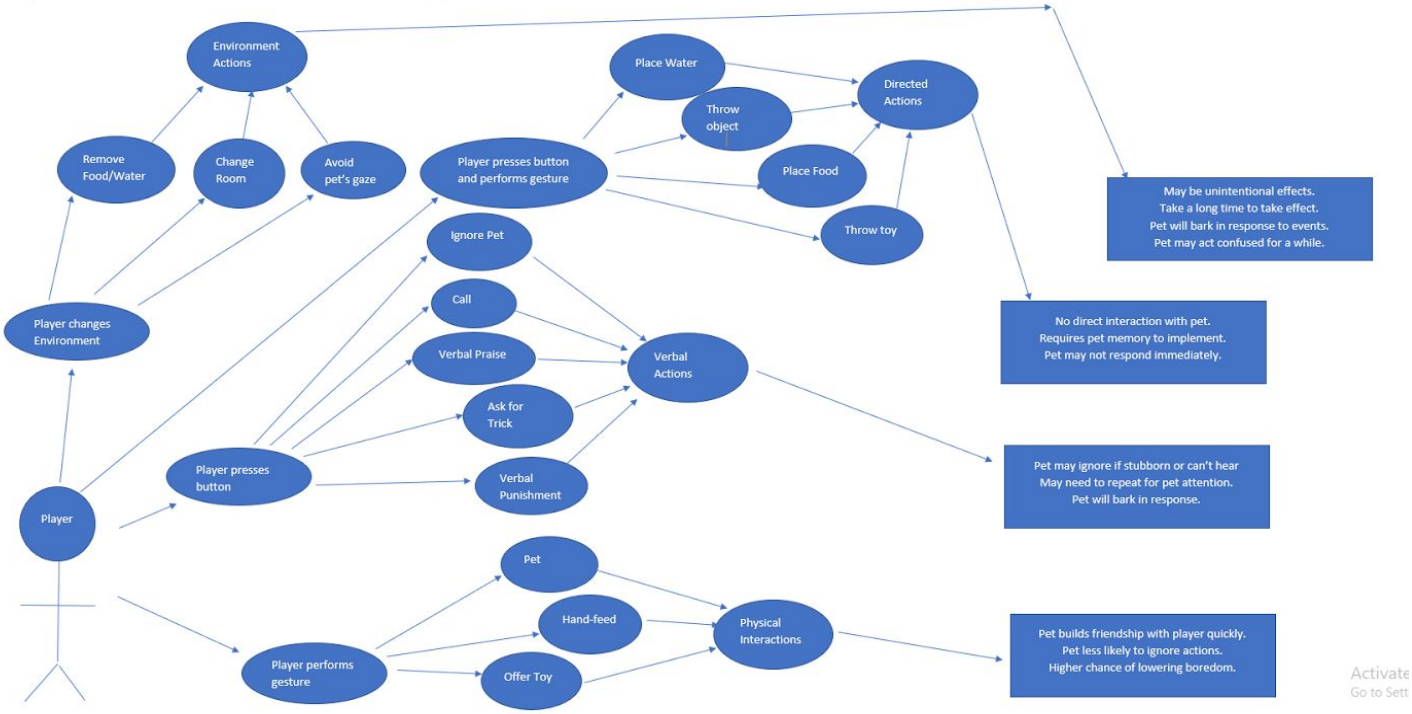
Github/Unity links

- https://github.com/Hansigzandrinovka/581-Ecosystem-Pet-Simulator
- Unity Cloud Project Name: EcosystemPetSimulator
- Unity Cloud Project ID: 4a251b7e-c15d-4ac6-9951-cdfa5f406abb

Final Project Design

Our project will consist of five components. Pet Design will govern the modeling, AI logic, and general actions performed by the pet. VR implementation will govern the Player's view of, and interactions with, the virtual world through the HTC Vive headset and its peripherals. The Game Model will handle the interactions of the other systems in the game, ensuring that the Pet, Player, and Environment can get all the relevant information they need from each other. The Environment will store the contents of the virtual room the player is in, and expose access to the player and pet to place furniture and located resources respectively within the room space. Finally, the User Interface will serve as the player feedback and tool of interaction with the environment. It will handle displaying to the player the pet's current highest need, and give the player a library of furniture and resource objects to place into the room.

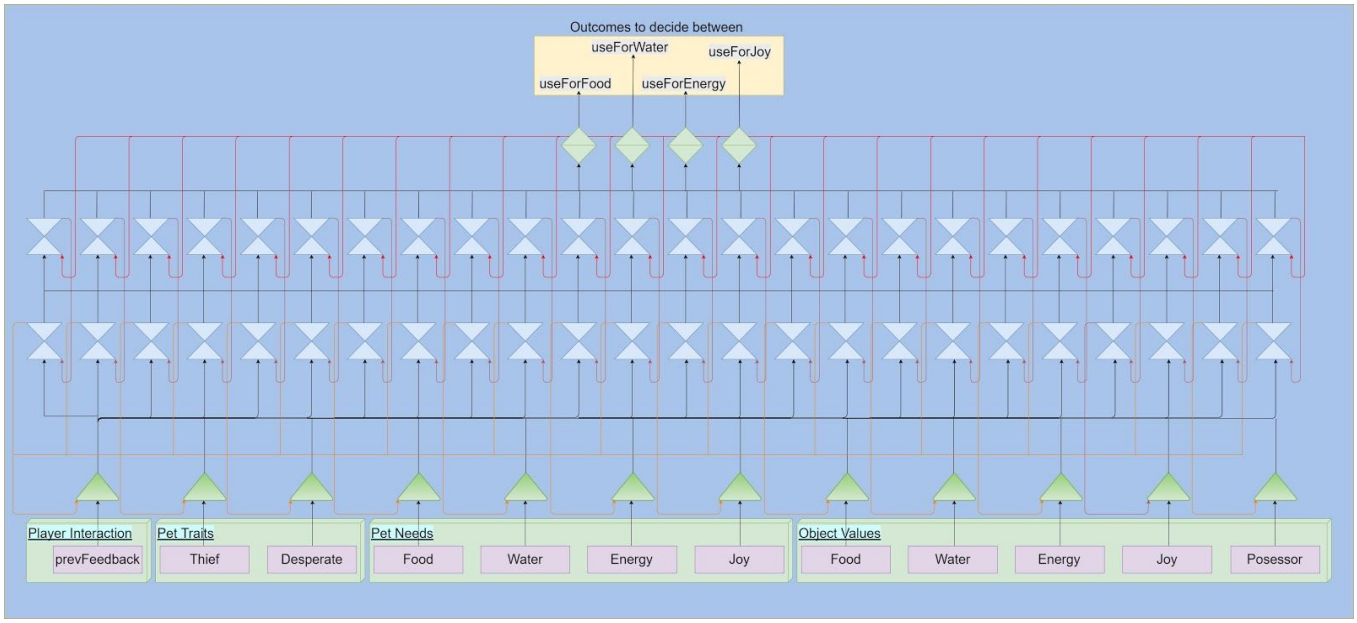
Pet Design



Player-to-Pet Use Case Diagram:
Download from
https://drive.google.com/file/d/0Bz17CpbcFn_bZ3RrR1lNSmxzNmM/view?usp=sharing
for a better view.

- Pet needs
    - Food - The pet will get hungry over time, more so when the pet is exerting itself, IE playing, running around. When the pet becomes very hungry, it will seek out any nearby food sources, or beg the player to provide one. After the player

provides a food source, the pet will try to eat from it or ask for food. Once the pet has brought its hunger-meter to near-empty, it will stop eating and re-evaluate its needs.
    - Water - The pet will get thirsty over time, more so when the pet is exerting itself. After the player provides a water source, the pet will seek it out. If there is no water inside, the pet will beg the player to fill it up. Then the pet will re-evaluate its needs.
    - Entertainment - The pet will get bored over time if not properly stimulated. The player will have a variety of toys they can use to play with the pet to keep it entertained. Unlike other needs, the pet will be almost always open to playing with the player if the player equips a toy. As boredom progresses in stages, the pet may ask the player to play with it, or find some other way to entertain itself.
    - Energy - Over time the pet will lose energy (at a variable rate depending on the action taken). Once it is depleted enough, the pet will look for surfaces/objects that are most beneficial to regain its energy tempered with the learned responses. For example, if the pet learns through feedback that a couch is not for sleeping, the pet should choose a safer (if lower value) object to sleep on to prevent the user's displeasure.
- Pet AI



    - This is the updated neural network framework. As can be seen, there are 12 float inputs on the bottom of the picture above. They get fed into the neural network input layer (the green triangles), where they are FedForward through the two hiddenLayers (the hourglasses). The two hidden layers is where the various trained weights of the neural network blend the 12 inputs to come out with 4 outputs (the yellow box on top), which the pet uses to decide which action to take for this object. Training works by during the appropriate phase, test inputs, and expected outputs will be given to the neural network. The neural network will adjust the weights by the use of the BackPropagation functions (the reddish arrows pointing away from the

layers to the previous layers) so that the error between what the neural network generates and the expected output is minimized.

- This is the algorithm that we will be using for the neural network.
- /*
- * INPUTS AS FLOATS, ARRAY PLACE GIVEN BEFORE TRAIT:
- * //FEEDBACK
- * 0 lastObjectFeedback: -1.0 to 1.0 (negative, zero, positive)
- * //PET TRAITS
- * 1 petIsThief: 0.0 to 1.0, reduces negative feedback, maxThief is 1
- * 2 petIsDesperate: 0.0 to 1.0, override negative feedback, maxDesperate is 1
- * //PET NEEDS
- * 3 petNeedsFood: 0.0 to 1.0 (1.0 is max Food need)
- * 4 petNeedsWater: 0.0 to 1.0 (1.0 is max Water need)
- * 5 petNeedsEnergy: 0.0 to 1.0 (1.0 is max Sleep need)
- * 6 petNeedsJoy: 0.0 to 1.0 (1.0 is max Joy need)
- * //OBJECT VALUES
- * 7 objectGivesFood: -1.0 to 1.0 (object causes hunger or cures it)
- * 8 objectGivesWater: -1.0 to 1.0 (object causes thirst or cures it)
- * 9 objectGivesEnergy: -1.0 to 1.0 (object causes tiredness or cures it)
- * 10 objectGivesJoy: -1.0 to 1.0 (object causes boredom or cures it)
- * 11 objectPossesor: 0.0, 1.0 (owner, pet)
- *
- * OUTPUTS:
- * 0 useForFood: Pet may decide to eat object
- * 1 useForThirst: Pet may decide to drink object
- * 2 useForEnergy: Pet may decide to rest with object
- * 3 useForJoy: Pet may decide to play with object
- */
-
- WHILE STARTING UP:
- RUN training using expected outputs, backpropagation, and selected inputs
- setLimitDontCare //(where we don't care about pet needs)
- setLimitDireCare //(where petNeeds are dire)
- setPetDecisionLimit[] //(where the pet will act or search for next object per need)
- WHILE INTERACTIVE WITH PLAYER
- //0--randomPetActs--limitDontCare--brainRunsPeriodically--limitDire--fixDireState--1
- GET current states of pet Periodically
- IF (FOR all pet states; all are less than LimitDontCare)
- random(idle, wander)
- ELSE IF (FOR all pet states; all are less than LimitDireCare)
- findAnObject
- SWITCH
- CASE: FOOD
- getLastObjectFeedBack

- getPetIsThief
- getPetIsDesperate
- new outPutsFromNN = brain.FeedForward(new float[] {inputs})
- IF outPutsFromNN[useForFood] >= petDecisionLimit[FOOD]
  - AlertPlayerAboutAction
  - open setPlayerFeedback for memOfPet
  - takeAction
  - close setPlayerFeedback for memOfPet
- ELSE
  - findNextObject
  - recursiveCall
- CASE: WATER
  - getLastObjectFeedBack
  - getPetIsThief
  - getPetIsDesperate
  - new outPutsFromNN = brain.FeedForward(new float[] {inputs})
  - IF outPutsFromNN[useForThirst] >= petDecisionLimit[THIRST]
    - AlertPlayerAboutAction
    - open setPlayerFeedback for memOfPet
    - takeAction
    - close setPlayerFeedback for memOfPet
  - ELSE
    - findNextObject
    - recursiveCall
- CASE: ENERGY
  - getLastObjectFeedBack
  - getPetIsThief
  - getPetIsDesperate
  - new outPutsFromNN = brain.FeedForward(new float[] {inputs})
  - IF outPutsFromNN[useForEnergy] >= petDecisionLimit[ENERGY]
    - AlertPlayerAboutAction
    - open setPlayerFeedback for memOfPet
    - takeAction
    - close setPlayerFeedback for memOfPet
  - ELSE
    - findNextObject
    - recursiveCall
- CASE: JOY
  - getLastObjectFeedBack
  - getPetIsThief
  - getPetIsDesperate
  - new outPutsFromNN = brain.FeedForward(new float[] {inputs})
  - IF outPutsFromNN[useForJoy] >= petDecisionLimit[JOY]
    - AlertPlayerAboutAction
    - open setPlayerFeedback for memOfPet

    - takeAction
    - close setPlayerFeedback for memOfPet
    - ELSE
    - findNextObject
    - recursiveCall
  - ELSE (a petNeed is dire)
  - AlertPlayer
  - DECIDE mostDireNeed
  - useClosetObject to fix dire need
- Pet Model
  - The player's pet will be a cat, though we are considering supporting other pets. The pet will support animations for walking, looking at the player, eating, drinking, and any interactions that the player can make with the pet, such as being patted, watching thrown objects, or being praised/punished by the player.

VR implementation

- For VR implementation, the Unity game engine makes things easy by rendering scenes to the VR headset automatically to where the user can look around. Scripts attached to the camera and to objects in the virtual world are used to handle interaction. For instance, when the user looks at an interactive item, a ray is cast from the eye to the object. The object detects the ray colliding with it and then may call a method to do something. Another script is used to detect other actions the user might perform, such as clicks, swipes, or trigger presses. For detecting the location of the user's controllers, Unity supports an api that allows us to read the input of each touch controller, and the HTC Vive's sensors and api will allow us to track the location of each controller to use with our software.

Game model

- The game model will abstract away the details of what happens when the player or pet perform particular actions or queries.
- When the pet looks for food sources, it will make a request for a resource object of type “food”, given its position, and the engine will perform the search, and return an optimal path.
- Pet, and possible player, movement will be governed with Controllers that support movement methods that will queue up the action to move to a particular object or location. The controller can return an event upon completion that the Pet or Player objects can then respond to with notifications or further actions. if the Pet needs to climb up something, the controller will determine if it can, and send a failure event if it cannot.
    - NOTE possible AI learning opportunity - there is a deviation range within which pet may “think” it can/not climb up something, where it will try to climb very cautiously, and give up after a certain failure count, or learn that it can climb/jump that height.
- The game will use Unity’s built-in physics system to support gravity, and physics-based object motion, but will selectively disable it to handle carrying and using of objects if we support the player doing hand interaction with the world.
- Primary Goals:
    - Pet will wander naturally when idle
    - Player can call pet to come closer
    - Player can play with pet through various toys and motions
    - Player can feed/water pet through placing objects in world
    - Pet will notify player of needs that cannot be met currently (missing water source, wants to play)
- Stretch Goals:
    - Player has some means of navigating room
        - teleportation
        - direct travel (may cause nausea/disconnect)
    - Player can earn income to buy toys and supplies for pet
    - Player can take pictures of pet
    - Player can take pet for walks
    - Multiple pets

Personalized Environment

- To create a more personalized experience for the user, there will be a portion of the user interface that allows players to select and place objects within the virtual space. The player will be able to reorganize their room in real time by dragging and dropping, while also able to place different objects like furniture and decor around their space. Making the user feel comfortable in a virtual space is a difficult task, especially since the user is wearing a bulky headset to immerse themselves in the world. The pet will be able to recognize where these objects are and will plan its route accordingly. At the beginning of

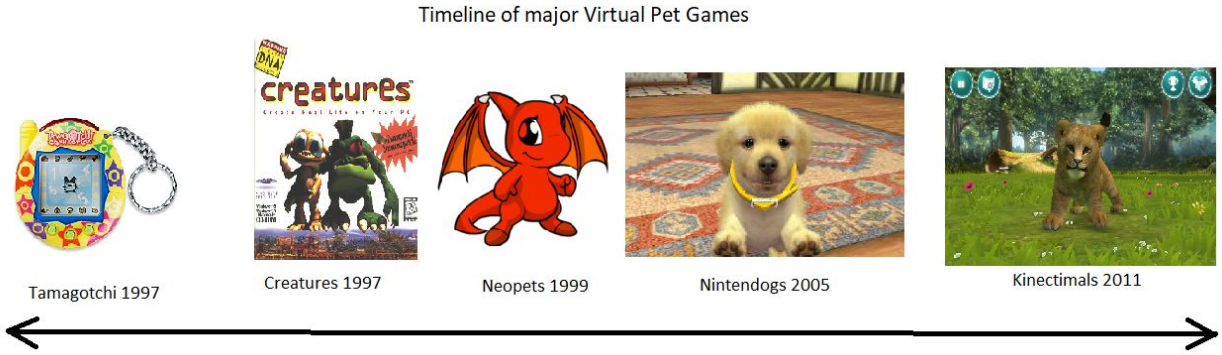
the game, the room will be a premade living room area and once the player has reorganized they can save their layout for future sessions.

- Pet Model(s)

The dog above is a placeholder taken from an open-source 3D modeling site, and was inspiration for the expected final quality of model to include in the game.

The cat above is the current pet model we are using for the project. Due to a hurried need to find open-source copyright-free models, a cat is our current pet model.

Similar AI and Pet Simulator Titles



- Our project aims to capture the successes of some of these games.
    - Tamagotchi introduced the idea of a pet that you had to take care of, and showed that pet simulators could be very successful games.
    - Creatures showed that it was possible for virtual creatures to be intelligent and adaptive organisms that learned from conditioning and experience.
    - Neopets created the model for pet games used today, where players accumulate cash that they spend to feed, take care of, and dress up their pet.

    - Nintendogs extended pet interactions to create a semblance of realistic pet interactions.
    - Kinectimals showed that virtual reality software (In this case, the Microsoft Kinect camera) was an effective medium to bridge the gap between virtual creatures and real people interacting with them.
    - We plan to create a game that uses VR and pet artificial intelligence to create an immersive pet simulator.

Ethical and Intellectual Property Issues

- Ethical Issues
    - Another concern of ours which we should be transparent about, is that this is a VR project, and, as such, all of the disclaimers associated with using VR fall on us as well. It is important that we also warn and make clear to the user all of the concerns that one would consider when using any VR technology, such as nausea, damage to surroundings, etc.
    - Another disclaimer is that the HTC Vive still utilizes wired technology, and so our project will require the user to remain in the area that the Vive's cord reaches.
- Intellectual Property Issues
    - The software that we are developing will be our own intellectual property, however, this will come with some caveats due to the usage of third party software and resources in our development process. One such caveat is that we are using the Unity Engine for our project, specifically Unity Personal, which requires that our software cannot generate an amount larger than $100,000 US in annual gross revenue or raised funds. This particular requirement will not be of too much concern to us, as we plan to advertise this software as free and open source.
    - As our software is original and not derivative of any other existing copyrighted software, our copyright rights and protections will lie with us as the owners of the original work, meaning that we alone are entitled to the reproduction, distribution, public display, public performance, and production of derivative works.
    - An intellectual property concern for us to keep in mind is that algorithms can be patented if it is embodied in a machine. Considering a sizeable portion of our project is AI, we will need to be careful to not infringe on the rights of any inventors of such patented algorithms.
    - Additionally, the persisting concern of software development is still important for us to bear in mind, and that is that our code should remain our own, and we should not plagiarize when we are developing our algorithms and implementations of our project.

Change Log

- Deleted mapping the physical world into the virtual world as both the Vive and the Oculus Rift do not have capability to take snapshots of the physical world
- Pet AI changed to be one larger neural network instead of a smaller neural network, decision tree, fuzzy state machine tied together
- Some tasks meant for winter break were unfinished, so were moved to the second semester.
- Due to potential copyright issues with Microsoft, the initial pet dog model was replaced with a cat model that is confirmed to be open-source.

Works Cited

Davis, Jeanie Lerche. "5 Ways Pets Can Improve Your Health." *5 Ways Pets Can Improve Your Health*. WebMd, 2004. Web.

"Healthy Pets Healthy People." *Centers for Disease Control and Prevention*. Centers for Disease Control and Prevention, 30 Apr. 2014. Web.

Oaklander, Mandy. "Science Says Your Pet Is Good for Your Mental Health." *TIME Health*. N.p., 6 Apr. 2017. Web.

Shatzman, Celia. "How Pets Improve Your Health." *Health.com*. N.p., 06 July 2015. Web.