Victoria Swartz
Assignment5
CSC 415

LuvU

https://github.com/ToriSwartz/LuvU

Use Cases:

Use Case: user feeds hungry virtual pet
Primary Actor: user
Goal in Context: increase the hunger meter of the virtual pet
Preconditions: user must be continuing a past game, hunger meter is low
Trigger: user selects food icon
Scenario:

1. User opens LuvU
2. User clicks the food icon
3. Apple appears on screen
4. User drags apple to virtual pet
5. Pet eating apple animation plays
6. Pet hunger bar increases set amount

Exceptions:

1. User does not drag the apple to the virtual pet; **user drops food**

Priority: High priority, implemented with basic functions.
Frequency of use: frequent

Use Case: user feeds full virtual pet
Primary Actor: user
Goal in Context: increase the hunger meter of the virtual pet
Preconditions: user must be continuing a past game, hunger meter is full
Trigger: user selects food icon
Scenario:

1. User opens LuvU
2. User clicks the food icon
3. Apple appears on screen
4. User drags apple to virtual pet
5. Pet refusing food animation plays

Exceptions:

1. User does not drag the apple to the virtual pet; **user drops food**

Priority: High priority, implemented with basic functions.
Frequency of use: frequent

Use Case: user drops food
Primary Actor: user
Goal in Context: increase the hunger meter of the virtual pet
Preconditions: user must be continuing a past game, hunger meter is full
Trigger: user selects food icon
Scenario:

1. User opens LuvU
2. User clicks the food icon
3. Apple appears on screen
4. User drags apple to virtual pet
5. User drops apple in area around pet
6. Apple falls and settles on ground

Exceptions: none
Priority: High priority, implemented with basic functions.
Frequency of use: frequent

Use Case: user puts the tired virtual pet to sleep
Primary Actor: user
Goal in Context: increase the energy meter of the virtual pet
Preconditions: user must be continuing a past game, energy meter is low
Trigger: user selects sleep icon
Scenario:

1. User opens LuvU
2. User clicks the sleep icon
3. Sleeping animation occurs for time based on bar percentage
4. Energy bar increases until it is full
5. Waking up animation plays

Exceptions:
1. The virtual pet is not sleepy; **user puts rested pet to sleep**

Priority: High priority, implemented with basic functions.
Frequency of use: frequent

Use Case: user puts the rested virtual pet to sleep
Primary Actor: user
Goal in Context: increase the energy meter of the virtual pet
Preconditions: user must be continuing a past game, energy meter is low
Trigger: user selects sleep icon
Scenario:

1. User opens LuvU
2. User clicks the sleep icon
3. Pet shakes head no

Exceptions: none
Priority: High priority, implemented with basic functions.
Frequency of use: frequent

Use Case: give virtual pet tablet
Primary Actor: user
Goal in Context: increase the happiness of the virtual pet
Preconditions: user must be continuing a past game
Trigger: user selects games
Scenario:

1. User opens LuvU
2. User clicks the game icon
3. User selects the tablet
4. Virtual pet watching tablet animation plays
5. Happiness bar increases

Exceptions:

1. Story event occurs; **tablet story event**

Priority: High priority, implemented with basic functions.
Frequency of use: frequent

Use Case: tablet story event
Primary Actor: user
Goal in Context: teach user about positive body image
Preconditions: user must be continuing a past game and have given the virtual pet the tablet
Trigger: random chance algorithm
Scenario:

1. User opens LuvU
2. User clicks the game icon
3. User selects the tablet
4. Virtual pet watching tablet animation plays
5. Happiness meter reduces
6. Dialogue comes from virtual pet
7. Module opens with 3 answers
8. User selects answer

9. Happiness meter changes based on answer
10. Module opens to explain how the user handled the situation

Exceptions: none
Priority: High priority, implemented with basic functions.
Frequency of use: infrequent

Use Case: send virtual pet to school
Primary Actor: user
Goal in Context: increase the happiness of the virtual pet by large amount
Preconditions: user must be continuing a past game
Trigger: user selects school
Scenario:

1. User opens LuvU
2. User clicks the game icon
3. User selects the tablet
4. Virtual pet going to school animation plays
5. Timer appears with countdown to return
6. Virtual pet returning from school animation plays
7. Happiness bar increases

Exceptions:

1. Story event occurs; **school story event**

Priority: High priority, implemented with basic functions.
Frequency of use: frequent

Use Case: school story event
Primary Actor: user
Goal in Context: teach user about positive body image
Preconditions: user must be continuing a past game and have given the virtual pet the tablet
Trigger: random chance algorithm
Scenario:

1. User opens LuvU
2. User clicks the game icon
3. User selects school icon
4. Virtual pet going to school animation plays
5. Timer appears with countdown to return
6. Virtual pet returning from school animation plays
7. Happiness meter reduces
8. Dialogue comes from virtual pet
9. Module opens with 3 answers
10. User selects answer

11. Happiness meter changes based on answer

12. Module opens to explain how the user handled the situation

Exceptions: none

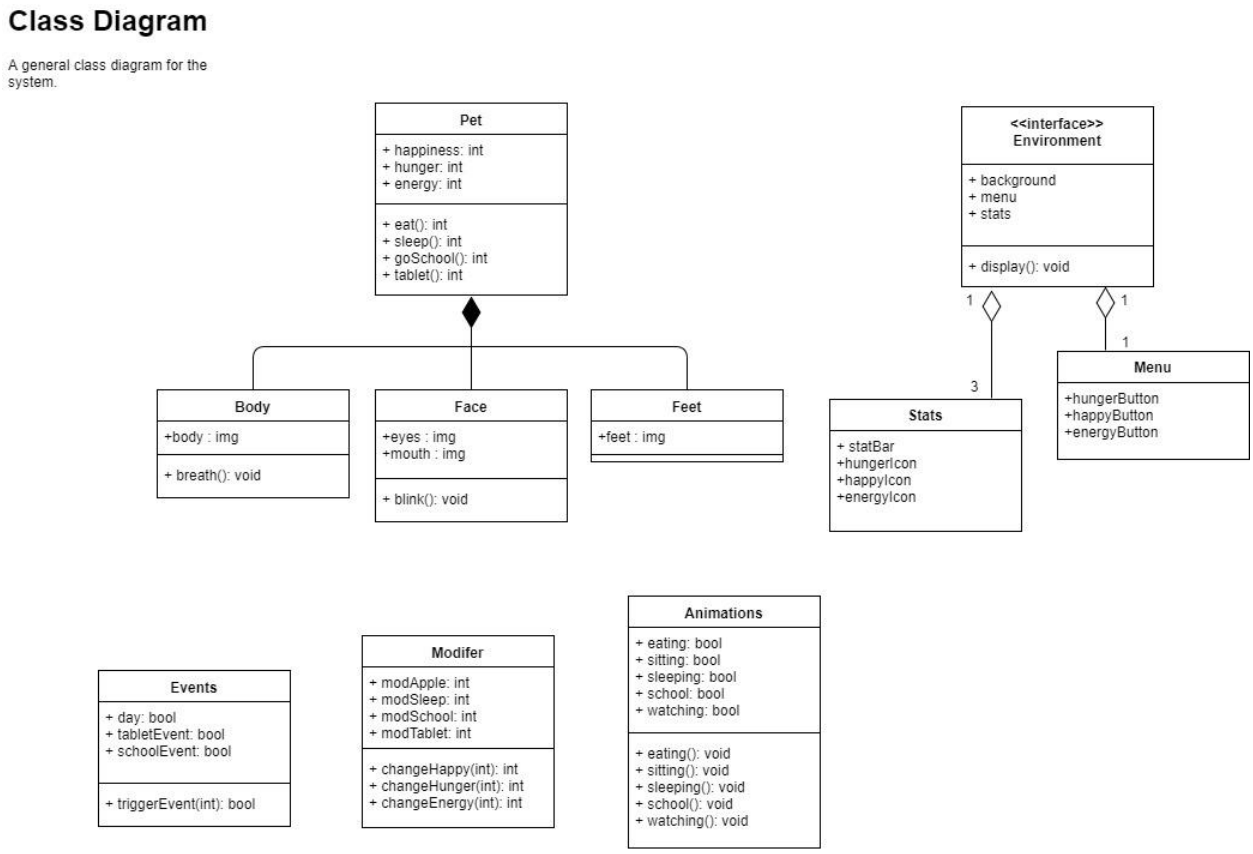
Priority: High priority, implemented with basic functions.

Frequency of use: infrequent

Class Diagram:

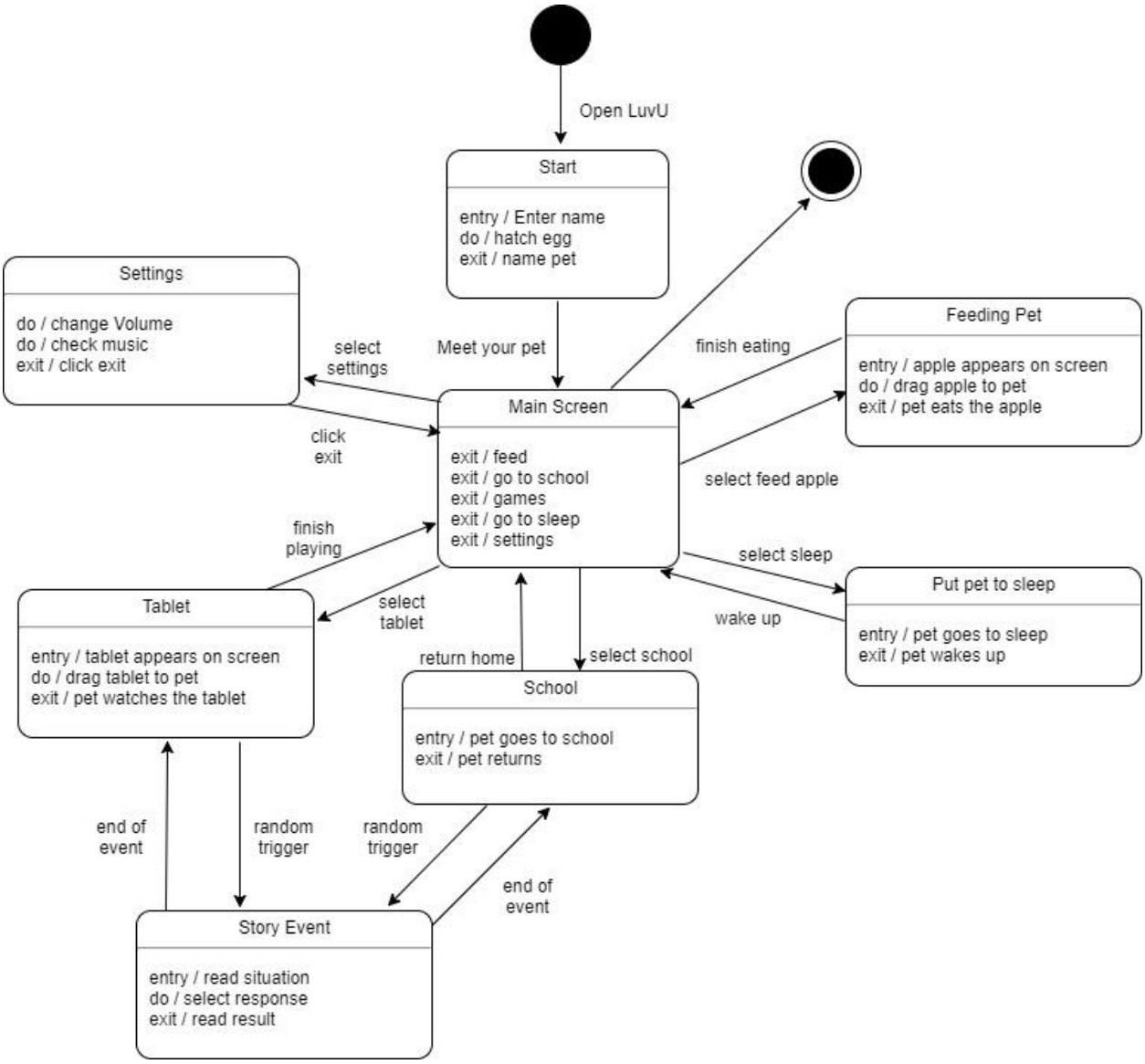
Victoria Swartz

Assignment5



State Diagram:

Victoria Swartz
Assignment5



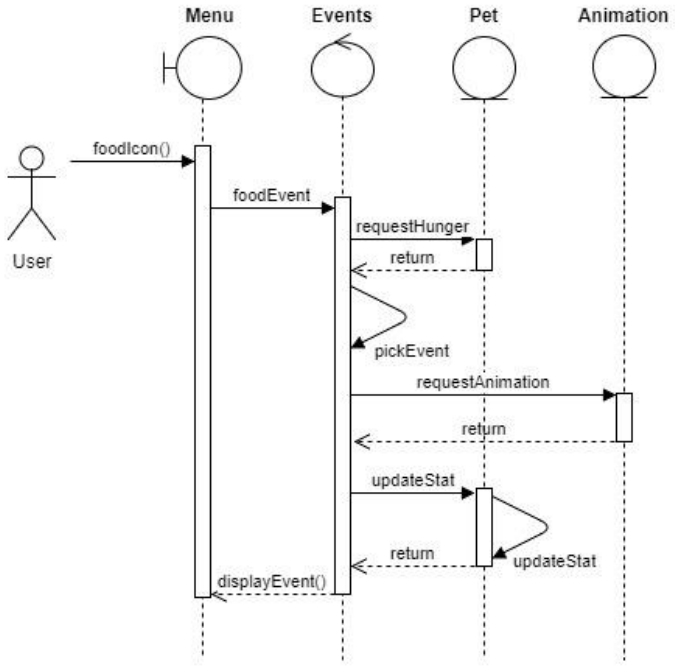
System Sequence Charts:

Victoria Swartz

Assignment5

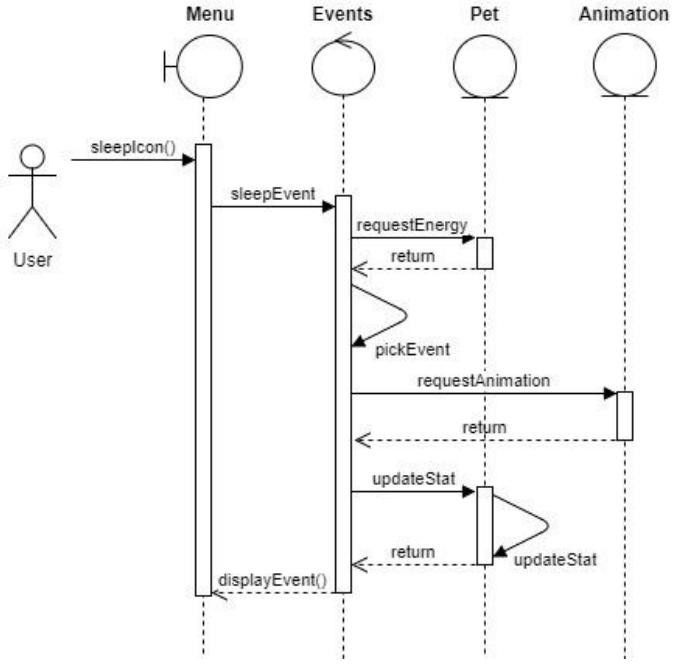
Feed Pet

The process of feeding the virtual pet.



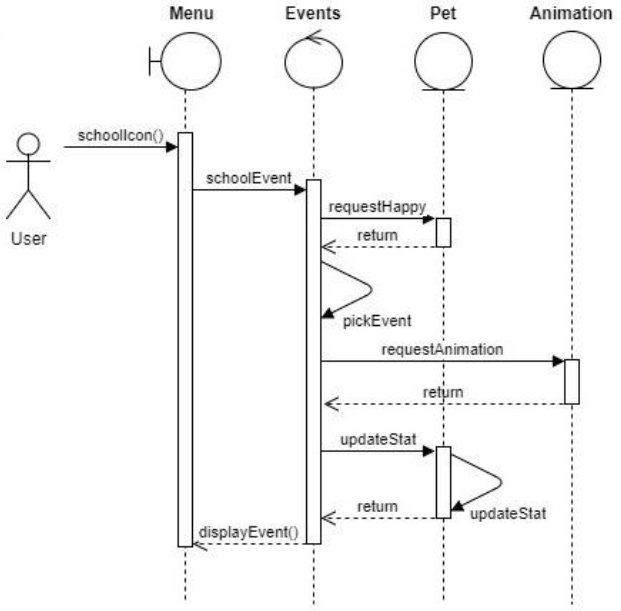
Send Pet to Sleep

The process of putting the pet to



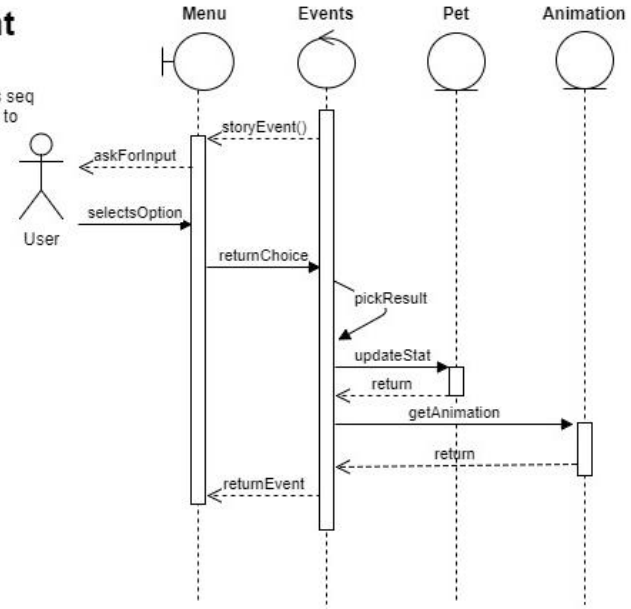
Go to School

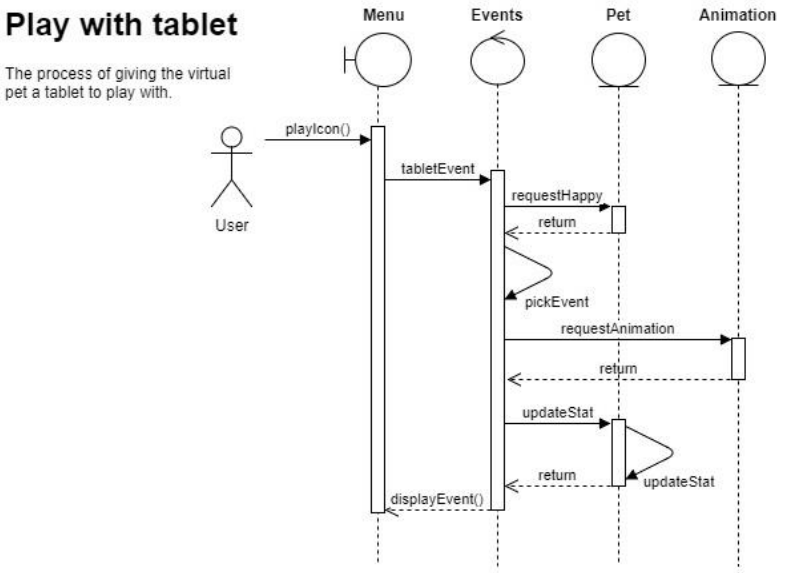
The process of sending the virtual pet to school.

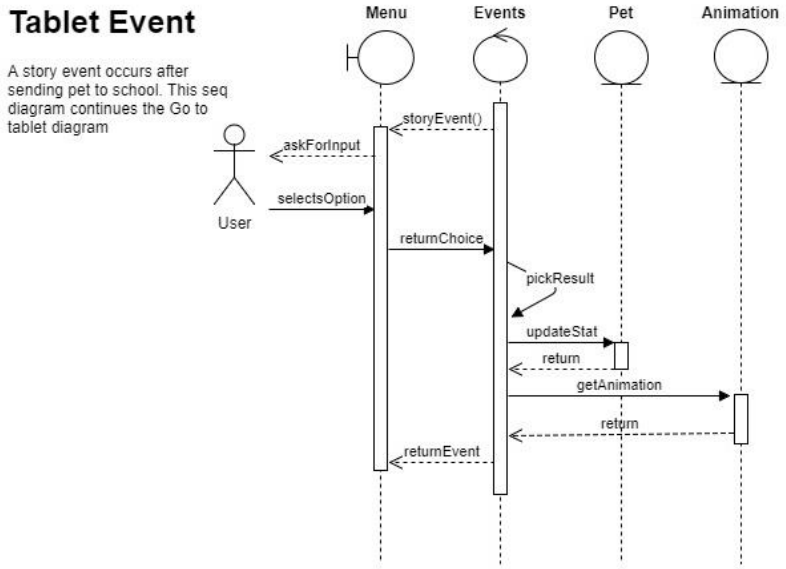


School Event

A story event occurs after sending pet to school. This seq diagram continues the Got to School diagram




Play with tablet
The process of giving the virtual pet a tablet to play with.
Menu
Events
Pet
Animation
User
playIcon()
tabletEvent
requestHappy
return
pickEvent
requestAnimation
return
updateStat
return
updateStat
displayEvent()


Tablet Event
A story event occurs after sending pet to school. This seq diagram continues the Go to tablet diagram
Menu
Events
Pet
Animation
User
storyEvent()
askForInput
selectsOption
returnChoice
pickResult
updateStat
return
getAnimation
return
returnEvent

UI Mock-Up:

1. **Strive for consistency.**
   Consistency will be easy to maintain throughout this project. Since there is only one main interactive screen the icons and stat icons will not be changing. Each Icon executes in a similar way. Once pressed an action occurs. The hardest part to keep consistent will be the pet animations since it must look like one character moving instead of many different frames of unrelated characters.
2. **Enable frequent users to use shortcuts.**
   This UI is pretty simple to begin with icons are all shortcuts to preform actions directly.
3. **Offer informative feedback.**
   Each button has an action that occurs that signals to the user that a button was in fact pressed beyond the button becoming darker as shown above. The apple button produces an apple in the bottom of the screen. The sleep icon produces a sleep action with dims the overall light of the scene and a sleeping animation takes place. The games button prompts a tablet to be given to the pet and an animation follows. School prompts the pet to get up and leave with additional sound queues and a time until they will get back. Setting will prompt a pop up.
4. **Design dialog to yield closure.**
   The best way to mark the end of the action will be a change in the stats at the top of the game. A text that will say +number or –number will be displayed at the end of every action along with a visual increase in the stat bar to give the user a sense of progress and closure.
5. **Offer simple error handling.**
   The errors the user can make include preforming an action on a pet that is not needed such as feeding a full pet. The program handles these errors with animations of the pet refusing to do the action so the user knows that they cannot perform the action.

6. **Permit easy reversal of actions.**
   All the actions are driven by the buttons in this interface. My design will cancel or stop and action if the darkened button is clicked again before the action is completed.
7. **Support internal locus of control.**
   It is the job of the user to fully care for this pet. So the increase of stats in completely dependent upon the user. They must click buttons to launch actions in order to care for the pet. Additionally, a lack of action will cause a visible reduction of stats and the pet will appear visually unhappy.
8. **Reduce short-term memory load.**
   The user will not be required to remember anything beyond what the 5 buttons do. This application is intended for kids so it is very simplistic. Each button also has a large icon that relates to its action as well once again limiting what the user must remember.

Test Cases:

Unit Testing: For my unit testing I will be using Visual Studio Testing framework. Each unit in my program will be a class. White box test cases will be written in order to ensure that each function is acting as intended. Since my team only consists of me at the very small unit test should make sure that before we integrate other parts of the application that the bare bones are structurally sound. If I implement a binary tree it might give me the right result but that does not mean its internal structure is sound.

Integration testing: This testing will also be done with the Visual Studio Testing framework. However, instead of white box testing these larger modules will be tested via black box testing. Since we know both classes are already structurally sound from the unit testing it is safe to say that errors can be attributed to miscommunication between classes which allows targeted debugging despite the black box aspect of the testing.

System Testing: System testing will best be handled by users testing the application manually. Automated testing is possible but it is also expensive in both time and software. It will be easiest to use my test cases to run through each scenario and make sure the application responds properly and allow users to make other comments about possible lag or other errors an automated machine may not pick up.

Functionality	Inputs	Expected Output	Output
user feeds hungry virtual pet	-click the food button -select the apple -drag it to pets mouth	-pet appears to eat apples -hunger increases	
user feeds full virtual pet	-click the food button -select the apple -drag it to pets mouth	-pet refuses to eat apple -apple drops and settles on screen	
user drops food	-click the food button -select the apple	-apple drops and settles on screen	

	-drops food not near pet		
user puts the tired virtual pet to sleep	-click the sleep button	-pet sleeps until energy == 100%	
user puts the rested virtual pet to sleep	-click the sleep button	-pet refuses to go to bed	
give virtual pet tablet	-click games icon -select tablet	-pet watches tablet -happiness increases	
tablet story event(Good choice)	-click games icon -select tablet	-pet watches tablet -pet asks user question -prompt appears -user selects answer -happiness increases a lot	
tablet story event(OK)	-click games icon -select tablet	-pet watches tablet -pet asks user question -prompt appears -user selects answer -prompt says how you could have handled this better	
tablet story event(BAD)	-click games icon -select tablet	-pet watches tablet -pet asks user question -prompt appears -user selects answer -prompt says how you could have handled this better -happiness decreases a lot	
send virtual pet to school	-click school icon	-pet goes to school -pet returns -pet says what it did at school -happiness increases a lot	
school story event(Good choice)	-click school icon	-pet goes to school -pet returns -pet asks user question -prompt appears -user selects answer -happiness increases a	

		lot	
school story event(OK)	-click school icon	-pet goes to school -pet returns -pet asks user question -prompt appears -user selects answer -prompt says how you could have handled this better	
school story event(BAD)	-click school icon	-pet goes to school -pet returns -pet asks user question -prompt appears -user selects answer -prompt says how you could have handled this better -happiness decreases a lot	

Modularity and encapsulation

Modularity of the program is shown through my organization of classes. Modularity can be shown through the class hierarchy for the pet as well as the environment. Moreover the system is broken up into many subsystems. The game will be made of a system to handle animation, events, stats, the pet and another will manage the environment. Each one will be made of smaller classes in order to maintain organization. Encapsulation is achieved by not allowing users to directly make any modifications to the variables. Additionally each class will have its own encapsulation. A good example is the Event class which will have many functions that each handle a different event, but due to their similar nature are grouped together.

Data structures and their appropriateness

LuvU will be responsible for managing 2D animations so the virtual pet can move, a system of emotions for the pet, rendering and destroying the graphics and managing user interaction during the story events. One of the issues will be managing the number of objects that will have to be rendered on the screen. These objects need to be created and destroyed constantly. For quicker access time I will be using a vector to store these objects to be rendered. This will allow me to use minimal space and add more objects as the project grows or as other people. The pet emotions may become complex with many facial expressions so a tree will be a good data structure to allow the game to quickly find the expression since rendering it will already take a substantial amount of time.

Elegance and Efficiency

My data structures and organization allow the game to be elegant and efficient. Vectors and binary trees both have a quick lookup time allowing the game to quickly grab and render scenes, but the overall structure also makes the game more efficient. The pet has its own series of classes so the pet can be animated without having to redraw the entire scene making lag less apparent. The event class will handle managing all the events which will also employ a tree like data structure so the proper event can be found quickly and it also allows a parent child relationship so cascading events will have very quick lookup times.