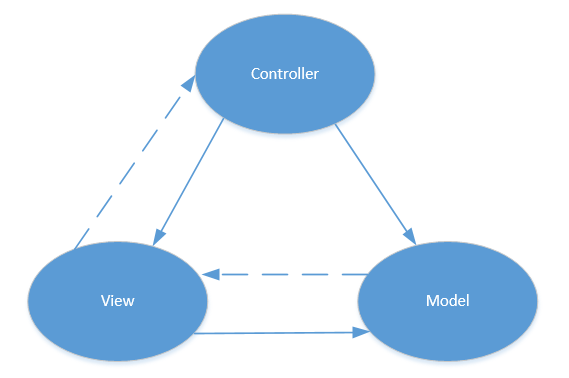
**High Level Architecture**

Architectural Pattern: Model-View-Controller

* Separates the representation of information from the user’s interaction with it.
* MVC will give our application data binding
  + Data binding establishes a connection between the User Interface and logic. When data is changed, the underlying data will reflect that change.



MVC

* Models represent application’s state.
  + Objects have properties and methods, so the application has state (the model).
  + The two main models will be the pet and its environment.
* Views are what the user sees and interacts with
  + Binds (Listens) to the models, so when the models change, the views update themselves.
  + They will display the pet and its environment.
  + In general, activities are the views.
* The Controller is the brains of the architecture.
  + Handles user interaction with the view
  + Interprets the user gestures and interaction
  + Controls the pet’s actions and feelings.
  + Updates the models, and sends messages back to the view.

Package Structure

Group classes together by their functionality

* Activities
* Controllers
  + Brains for the views
  + Controls how the pet reacts to every situation
* Lists (Custom list adapters)
  + List of accessories
  + List of play options
  + List of pet cleanliness
  + List of pet facial expressions
  + List of pet colors
  + List of skies
* Models

Interfaces

1. Pet’s Home
   1. This is where the pet lives. Here, the user interacts with the pet in many different ways.
   2. Buttons are used to do various things

<<Omar’s stuff here>>

<< Pictures >>

**Class Documentation**

1. Pet – contains information about the user created pet.
   1. Fields
      1. String petName – name of stored pet.
      2. Int width – width of pet image (varies based on screen size).
      3. Int height – height of pet image (varies based on screen size).
      4. Float xCoordinate – current top left x coordinate of pet image, is updated accordingly.
      5. Float yCoordinate –current top left y coordinate of pet image, is updated according.
      6. Int hungerLevel – int value up to 100. Denotes hunger of pet in a percentage.
      7. Int happinessLevel - int value up to 100. Denotes happiness of pet in a percentage.
      8. Int energyLevel - value up to 100. Denotes energy of pet in a percentage.
      9. Boolean isSleeping – Boolean to check if pet is currently asleep.
      10. Boolean isEating – if true, eating animation will occur.
      11. Boolean needFood – if true, a thought bubble with corresponding image will appear by pet’s head
      12. Boolean needBathroom – if true, a thought bubble with corresponding image will appear by pet’s head.
   2. Methods
      1. None
   3. Constructor
      1. Default
2. Startup Activity – occurs when the user first starts the application
   1. Fields
      1. None
   2. Methods
      1. Protected void onCreate (Bundle savedInstanceState) – check if pet exists on user’s phone. If it does, go to HomeActivity else allow user to create a pet via CreateActivity.
      2. Public Boolean onCreateOptionsMenu(Menu menu) – Inflate the menu, this adds items to the action if it is present.
      3. Public Boolean checkForPreviousPet() – check preference file for existing pet.
   3. Constructor
      1. Default
3. HomeActivity – the main screen of the application.
   1. Fields
      1. ImageButtons – Buttons on the bottom of screen that allows the user to do various actions. These include playButton, acessorizeButton, poopButton, feedButton, and cleanButton.
      2. Pet pet – the stored pet object.
      3. Private HomeView hView – the bitmap of the background.
      4. Window parameters – These include windowWidth, windowHeight, playgroundHeight, playgroundWidth, and menuHeight. All are integers.
      5. Fields used to check if user has clicked on pet or not. These include float mActivePointerId, int mLastTouchX, int mLastTouchY, int mPosX, int mPosY,Boolean petIsClicked.
   2. Methods
      1. Protected void onCreate(Bundle savedInstanceState) – calculate screen metrics, draw bitmap background, draw pet, draw image buttons, check if pet or any button is pressed and take appropriate action.
      2. Public Boolean onTouch(View v, MotionEvents event) – calculate where user clicked, save user’s last click, call method isPetClicked to check if pet is clicked.
      3. Protected void onPause() – save pet information to preference file.
      4. Public Boolean onCreateOptionsMenu(Menu menu) – Inflate the menu, this adds items to the action if it is present.
      5. Public void savePetLocation() – save pet location to preference file, so that the pet remains in that location the next time the application is started.
      6. Public void isPetClicked(float userX, userY) – check if user click is within the area of the pet.
      7. Public void feedPet() – if feed pet button is pressed, this method will decrease pet hunger to 0%. Every minute pet hunger increases by 1%. If pet is at 100% hunger for more than 8 hours, it will run away. After eating, pet will generate random amounts of poop on the screen.
      8. Public void removePoop() – User needs to toggle the scooper button. Once toggled, the user can click and fling off screen the pet’s waste. User must click the scooper button again to take any other action.
      9. Public void bathePet() – If user clicks the clean pet button, the pet will be washed and gained happiness. Pet’s are fickle beings and will only tolerate being washed once a day.
      10. Public void playWithPet() – If user click play with pet button, the user can play fetch with the pet. This will increase happiness but decrease energy. Energy comes back at a rate 1% per minute up to 100% as a maximum and 0% as a minimum.
      11. Public void runawayPet() – if pet is ignored for too long, it will run away.
      12. Public void notifyOwnerOfNeglect() – notification to owner that pet is close to running away.
   3. Constructor
      1. Default
4. HomeView – sets background and pet images
   1. Fields
      1. Private Bitmap mBoard – background image of application to be drawn.
      2. Private Bitmap mPet – pet image to be drawn.
      3. Private Point petPoint – current location of pet, in terms of x and y coordinates.
   2. Methods
      1. Protected void onDraw(Canvas canvas) – set the layout parameters, build board and scale it to the screen size. Draw background image/
      2. Public void drawPet(float x, float y) – takes in beginning pet location, calls onDraw() to redraw board and add the pet for the first time.
      3. Public void dragPet(float x, float y) – takes in new pet location and calls onDraw() to redraw board and pet each time the pet is moved.
      4. Public void drawMenuButtons() – on the button of the screen place the five image buttons used for interaction with the pet.
   3. Constructor
      1. HomeView(Context, AttributeSet) – constructor for the background image.
5. AccessorizeActivity – brings up a different screen where the user can select accessories for their pet.
   1. Fields
      1. Bitmap hat – the hat the pet is wearing, defaults to none.
      2. Bitmap shirt - the shirt the pet is wearing, defaults to none.
      3. Bitmap misc – misc accessories held in hand of pet, defaults to none.
      4. ImageButtons – each accessory will have its own image button that the user can press to add that accessory to their pet.
      5. Bitmap background – solid color background for the menu.
   2. Methods
      1. Public void addHat() – User can select from various hats to add to their pet. This method will draw the pet with the selected hat until the user selects no hat from the accessory menu.
      2. Public void addShirt() - User can select from various tops to add to their pet. This method will draw the pet with the selected top until the user selects no top from the accessory menu.
      3. Public addMisc() - User can select from various miscellaneous items to add to their pet. This method will draw the pet with the selected items until the user selects no misc from the accessory menu.
   3. Constructor
      1. Default
6. PlayActivity – mini games that the user can play with their pet.
   1. Fields
      1. None
   2. Methods
      1. Public void selectGame() – User selects which game to play with their pet.
      2. Public void startGame1() – User plays game one with pet.
      3. Public void StarGame2() – User plays game two with pet.
      4. Public Boolean onCreateOptionsMenu(Menu menu) – Inflate the menu, this adds items to the action if it is present.
   3. Constructor
      1. Default

**Requirements Trace**

|  |  |
| --- | --- |
| Previously Specified Requirement | Requirement Trace |
| The app shall check the device for previous instances of a virtual pet. | This requirement will be satisfied if the app detects a previous pet file in the expected directory. |
| The app shall allow the user to choose the color of the eggs he/she will hatch, as well as the gender and the species of the hatched pet. | This requirement will be satisfied if the app requests the gender and egg color information from the user and said information persists through the lifespan of the pet |
| The Virtual pet shall age from an infant to an adult, with the transition taking place after one week. | This requirement will be satisfied if the pet linearly ages within the span of one week and remains in adult form through its lifespan. |
| The virtual pet shall threaten to run away if the user neglects it for two days. The user will have 36 hours to respond after the warning is sent. | This requirement will be satisfied if the app successfully conveys the pets sincere intent to leave forever at 48 hours after the last attention the pet was given. |
| The virtual pet shall “run away” if neglected for 48 hours plus 36 hours after the user has been notified of neglect | This requirement will be satisfied if the pet runs away after a total of 84 hours of neglect. At this point the user can no longer access the pet within the confines of the app. |
| The app shall allow the user to feed the virtual pet when it is hungry. | This requirement will be satisfied if the app indicates that the pet is hungry to the user and if upon feeding the pet the hunger levels are reduced. |
| The app shall allow the user to “pet” the virtual pet at any time. | This requirement will be satisfied if the app at any non menu state, permits the action of petting the pet which must improve the happiness of the pet. |
| The virtual pet shall sleep when the energy level is low. | This requirement will be satisfied if the app puts the pet into a sleep state after a pretedermined threshold of sleepiness due to play and lack of sleep is crossed. |
| The virtual pet shall have a though bubble that helps the user provide care. | This requirement will be satisfied if thought bubbles appear periodically in sufficiently entertaining intervals. The thought bubbles must convey information about the pets desires. |

**Trace of Interface Requirements**

The interface requirements will be met if the app accepts user input, stores data to phone memory with 1 minute of temporal resolution and can propagate messages to the operating system.

**Trace of Physical Environment Requirements**

The application will meetthe physical environment requirements if it runs reasonably on a Samsung Google Nexus S. While some choppy performance may be tolerated the app must be able to function.

**Trace of Users and Human Factor Requirements**

The Human Factor requirements will be met if the app is sufficiently simple and users can understand the general functionality after a few minutes.

**Trace of Data Requirements**

The data requirements will be met if the app generates all appropriate events within a minute and the last critical interaction is recorded.

**Trace of Security Requirements**

The security requirements shall be met if the app can access files on the android device, backup its own data reliably upon exit, not be permitted to tamper with other applications and not report the user's personal info to eastern European crime lords.

**Trace of Quality Assurance Requirements**

The quality assurance requirements will be met if the app does not continuously crash. The app should boot in a reasonable amount of time. The app will run reasonably on Android 4.3 or later.