CS3217
Problem Set 3
HuffPuff Game

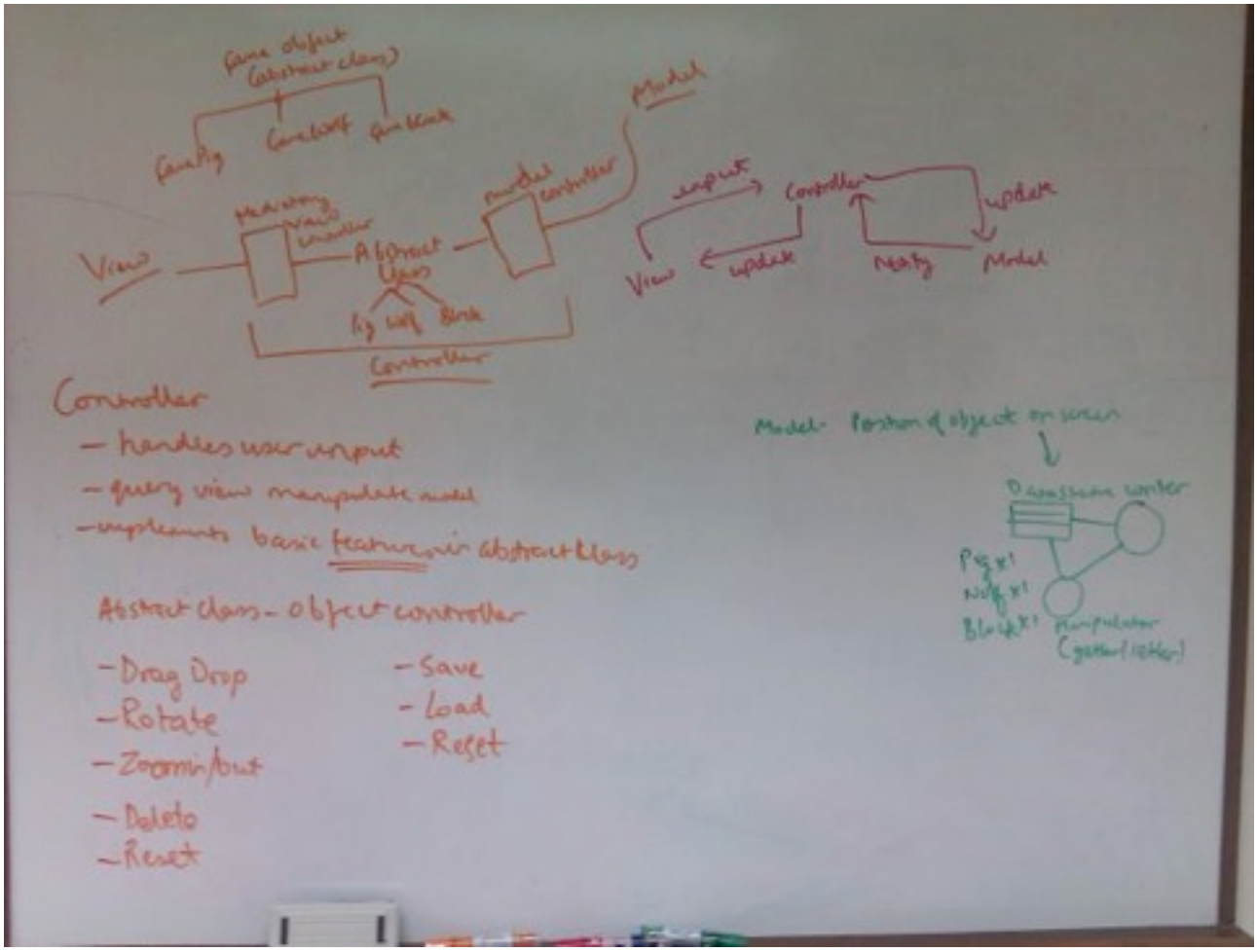
Angad Singh
U099118R

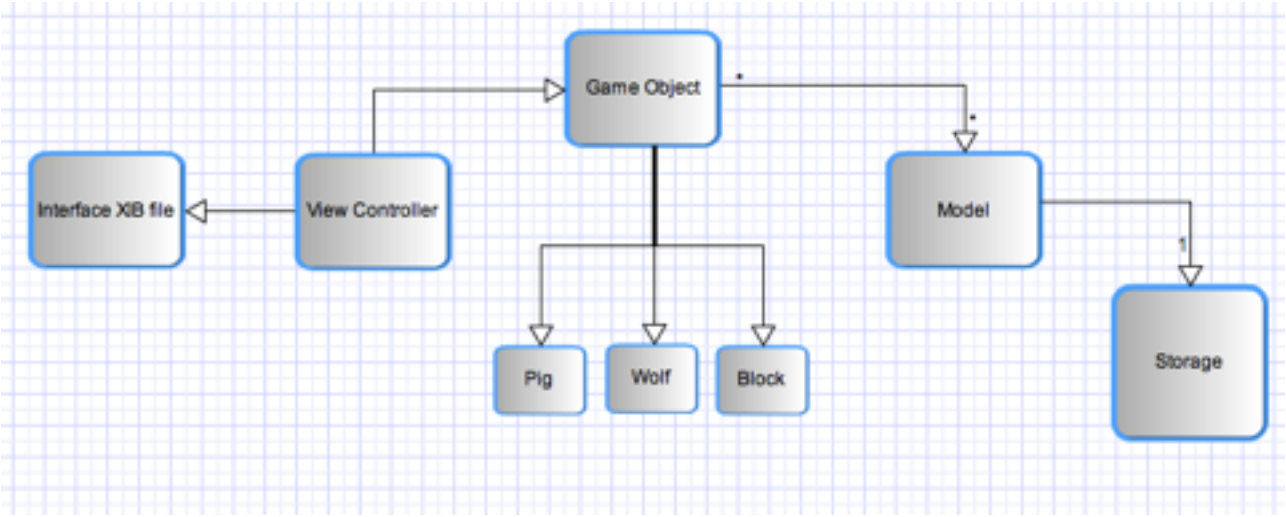
Questions and answers

1)MVC

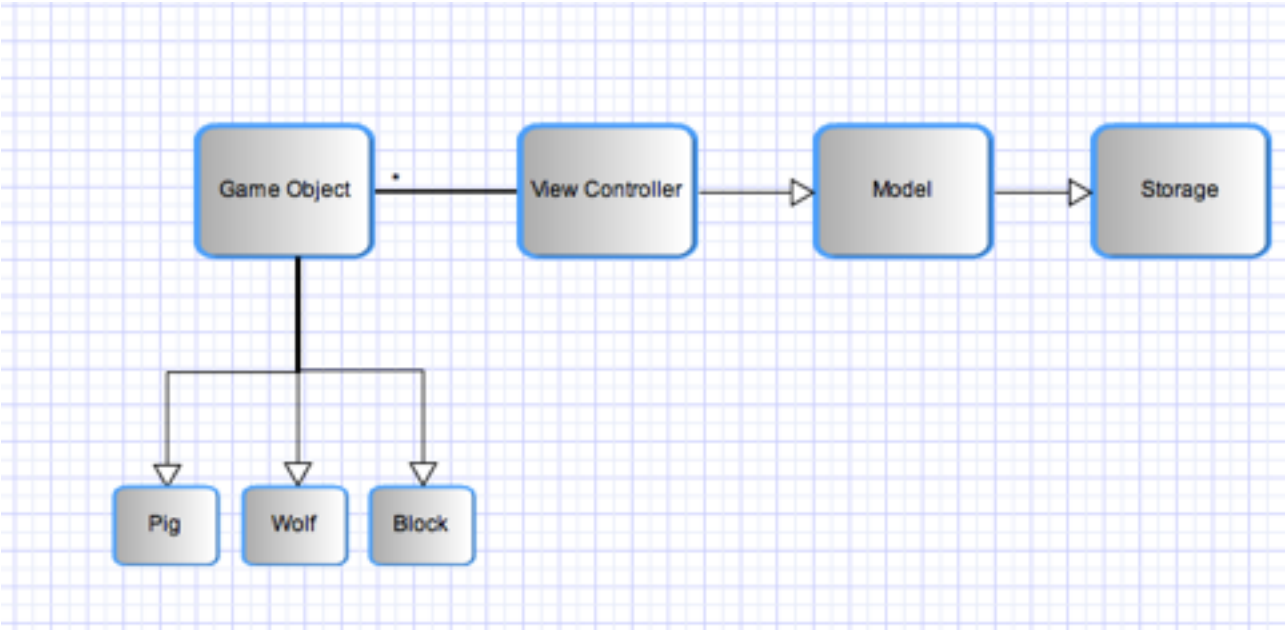
Implementation of -
- ViewController
- Game Object
- Pig, Wolf and Block
- Model (Data Structure)
- Storage (File handler)

This is what I wanted my code to be organized as -





And this is what the design currently



I can definitely bridge the gap between the two designs. But there were some issues I faced as I tried to get close to my earlier design. Some things are really a pain in Objective C. I have listed the issues that I faced here -

The major problem faced with the current implementation was that the Gesture listeners (Gesture recognizers) methods could not be delegated to methods that were outside the ViewController class. Do let me know if there is a way to do that and I will definitely improve my design.
Next is that the 3 subclasses (Pig, Wolf and Block) could not store their objects in a central model. A workaround was that there would be one instance of the model in the controller which would be then passed around to the pig wolf and block. But this implementation got me scratching my head for 2 days. There were some random bugs and I could not maintain a central copy of the objects array.

Current design description -
MVC Model View Controller
Model - NSMutableArray of the UIImageView of gamearea subviews and NSMutableArray of UIImageView of palette subviews.
View - The View here is the XIB interface builder file.
Controller - GameObject and its subclasses - Pig, Wolf and Blocks. They get the information from the view controller and implement the various gestures of the view objects.
ViewController - It is the central controller that gets information from the view and asks the model to store the information.

The MVC model that I have implemented here is rather a loose MVC model in which there is a controller that interacts with the View and with the model (called the View Controller and the Model Controller).

2) Alternative designs discuss

The one loop hole with the current implementation is that the view is closely related to the model in the sense that the Game Objects are being stored as they are in the model.
The reason why I decided to go ahead with my current implementation is that it was really difficult to maintain a central copy of all the objects (to be stored) as the Pig, Wolf and Block were 3 different subclasses.

Another possible alternative design could be that the GameObject implements a central controller class and there are multiple instances of the GameObject. Instead of the Pig, Wolf and the Block classes inheriting from the central Game Object.

The design 1 that I have given on the previous page is what can be a possible alternate design.

3) Wolf breath - projectile object

From my current understanding the projectile will be another subclass of the GameObject. It will be added to the gamearea on a particular event (Tap on wolf). It can be multiple objects as well, just to increase the physical realism of the wolf breath.

The projectile motion of the breath can be described in the tap event. Or to make the design more strong, the projectile motion itself can be designed in the physics engine.

4) Physics engine integration

The physics engine can be introduced as a controller. Basically it can provide a library of functions to the Game Object. It could take in the Game Object and redirect information to the view according to the functions inside the physics engine.

5) Functionalities, extend code
The other functionalities that I can think of right now for the Game - addition of more pig objects. Just like angry birds has many pigs, this game can have multiple pigs. The functionality can be introduced just as there can be multiple blocks. There would be some tweaking up of the code here and there.

Similar to the Wolf breath, other game objects such as fireball, wolf spit can be added. They can be added as Game Objects and their behavior can be controlled and defined by the Physics Engine.

Another functionality which may be helpful for the game is the ability to change the background. Each level could have a different background. The different backgrounds could be implemented in the palette and could be selected.

Testing

The game is mostly tested by loading on the iPad and testing by playing. Many small bugs were discovered and rectified.
To implement various methods of testing -

Black Box testing -
Implemented by testing the various elements from the UI and creating a bug list.

- Game Objects
    - Tap
    - Rotate
    - Drag and Drop
- Features
    - Save
    - Load
    - Reset

Glass Box Testing
Implemented by testing the internal functions such as writeToFile. It basically goes one level deeper than the Black Box testing

- Game Objects
    - Drag and Drop
        - Translate function NSLog the coordinates
    - Rotate
        - NSLog the coordinates and angle and verify
    - Tap

- Features
    - Save
        - Test the Array of objects
        - Check if the file writes correctly
    - Load
        - Test the objects loaded from the file

Saving and Loading the Game

The current implementation of saving the game is not what had been planned. But some stupid bugs cropped up while I was trying to implement the model for the game object. I spent 2 days trying to debug but in vain. So ultimately my current Save game feature works this way -
I am currently saving data using Object archival using the NSKeyedArchive. I take the gamearea from the view and add all the subviews of the gamearea to the model (which is an NSArray). The UIViews are then stored in the file using NSKeyedArchiver. The UIImage

does not conform to NSCoding so it is encoded as NSData in the UIImageExtensions class.

The file is read and the data is loaded inside the model's array. The UIViews have been assigned a tag to describe what class they belong to. All the gamearea and palette objects are removed and the ones loaded from the array are added to the gamearea and palette. New objects for Pig, Wolf and Block are created so that proper gesture handlers can be attached.

The design currently does not have the feature of saving multiple files. Internally, multiple files can be saved and loaded. I tried incorporating a UIPickerView to load multiple files, but I could not get it to load the files then. Will integrate the UIPickerView in the next iteration.

--I had started this assignment on Friday night. Well, if I had more time, I could have spent more time on fixing the bugs.
Personally, I think that the time given for this assignment was a bit less. The Cocoa framework can be nasty sometimes and it takes a long time to figure out the bugs without proper debugging in XCode. Anyway, will improve in the next iteration. Some small bugs which I know about are already there.