2014

Milestone 3

SDV602

REBEKAH ROSSITER

Contents

Introduction

This report contains all three milestones for SDV602. It contains all of the processes that I have gone through to complete my game. This has been requested by Todd Cochrane and is due on Friday 21st November at 9 am. I have however received an extension until Friday 28th November at 9 am. I have added some commenting to milestone one and edited code in milestone two.

Milestone One

Introduction

Who Killed Mr Sims is a text based game that is based on a solving a murder case. The purpose of this report is to show my understanding of the processes that I have undertaken in creating this game. As the project is indicative this is the first step of my work. This has been requested by Todd Cochrane and is due on Friday 29th August at 5 pm.

Findings

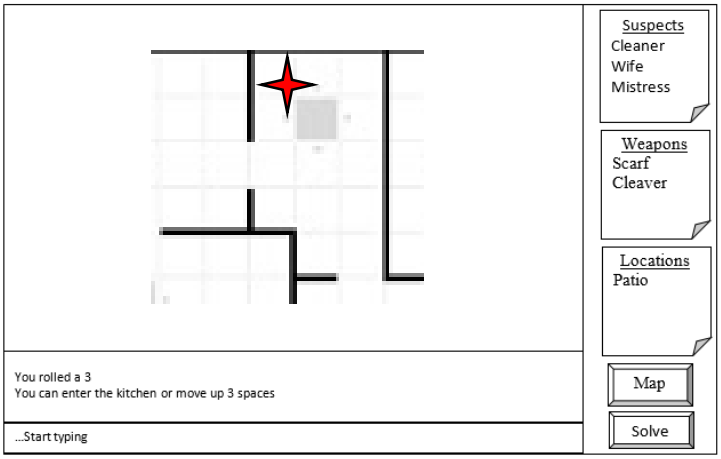
Project Brief

My game is called Who Killed Mr Sims. It is going to be a text based interactive game. The aim is to work out who killed the victim, where the victim was killed and what weapon was used. The victim's name is Walter Sims. There are 5 suspects that are: his wife, his mistress, the cleaner, the gardener and his neighbour. There are 5 locations that are: the bedroom, bathroom, lounge, kitchen and patio. There are 5 weapons which are: a knife, gun, scarf, cleaver and poison.
The main screen will show the map with a little list of suspects, weapons and locations. As the player goes through the game he/she can go into those lists and remove items off of those lists. When the player thinks that they have the answer they can solve it to see if they were right. There will be a section at the bottom of the page that allows the player to type and see what they have previously typed. That is also where any information from the game will be displayed.
When a player types roll and presses enter they will get a random number of spaces to move and options of what direction that they can go in.

Storyboards

Screenshot 1

This is the main screen with example data
There are;
> 3 lists of evidence *Screenshot 1*
> two "buttons" indicating a map and solve
> a textbox *Screenshot 3*
> a box for the "conversation" *Screenshot 4*
> players current position on map *Screenshot 11*



Screenshot 2

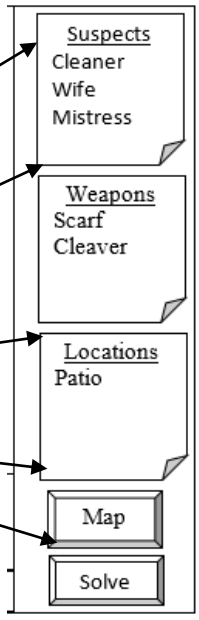
This contains a list of current suspects. You can open it to view the suspects that you have ruled out. *See screenshot 5 for expansion of list.*

This contains a list of current weapons. You can open it to view the weapons that you have ruled out. *See screenshot 6 for expansion of list.*

This contains a list of current locations. You can open it to view the locations that you have ruled out. *See screenshot 7 for expansion of list.*

This takes you to a bigger view of the map. *Screenshot 10*

This ends the game and shows whether you solved the case or not. *Screenshots 8 & 9*



Screenshot 3

This is where the user types text. After typing the user must press enter.

...Start typing

Screenshot 4

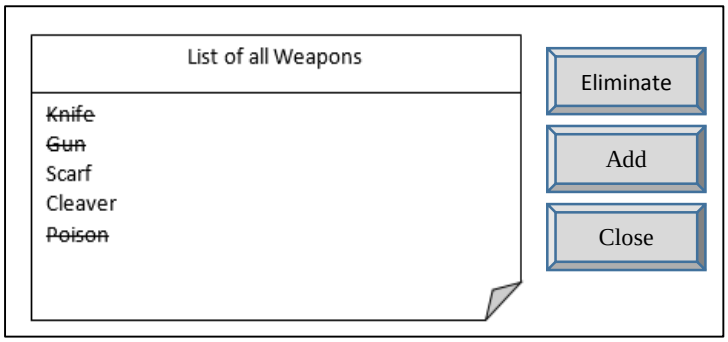
This is where the text that the user entered shows up and what the games reply is.

You rolled a 3
You can enter the kitchen or move up 3 spaces

Screenshot 5

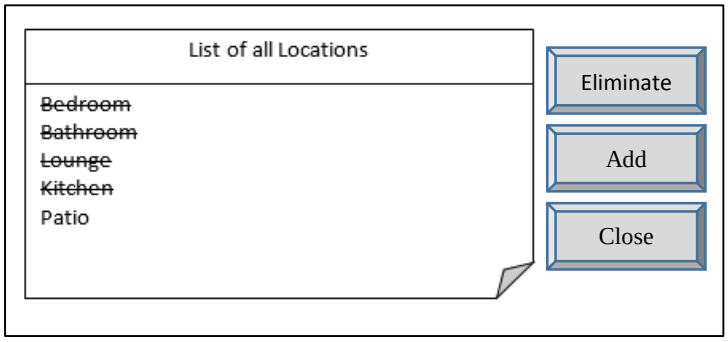
This will open on top of the map. It is an example of the complete suspect list. It has two suspects with a line through their names as they have been eliminated as suspects. There are three options that are eliminate, add and close. Eliminate puts a line through a name, Add removes the line from a name and Close closes the view.

Screenshot 6



This will open on top of the map. It is an example of the complete weapon list. It has three weapons with a line through their names as they have been eliminated as the murder weapon. There are three options that are eliminate, add and close. Eliminate puts a line through a name, Add removes the line from a name and Close closes the view.

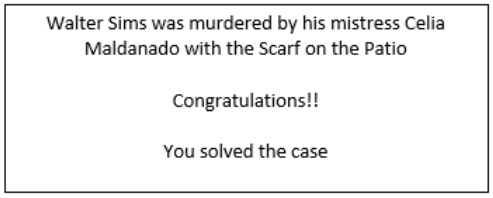
Screenshot 7



This will open on top of the map. It is an example of the complete location list. It has four locations with a line through their names as they have been eliminated as the murder location. There are three options that are eliminate, add and close. Eliminate puts a line through a name, Add removes the line from a name and Close closes the view.

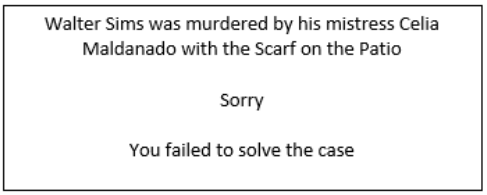
Screenshot 8

This is the message that will pop up if the player had the correct suspect, weapon and location



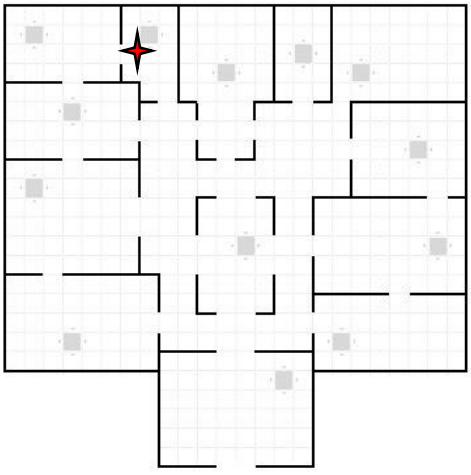
Screenshot 9

This is the message that will pop up if the player had the incorrect suspect, weapon and location



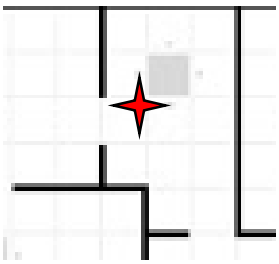
Screenshot 10

This is the complete map. The star is a pointer showing the players position on the map. *This is just an example.*

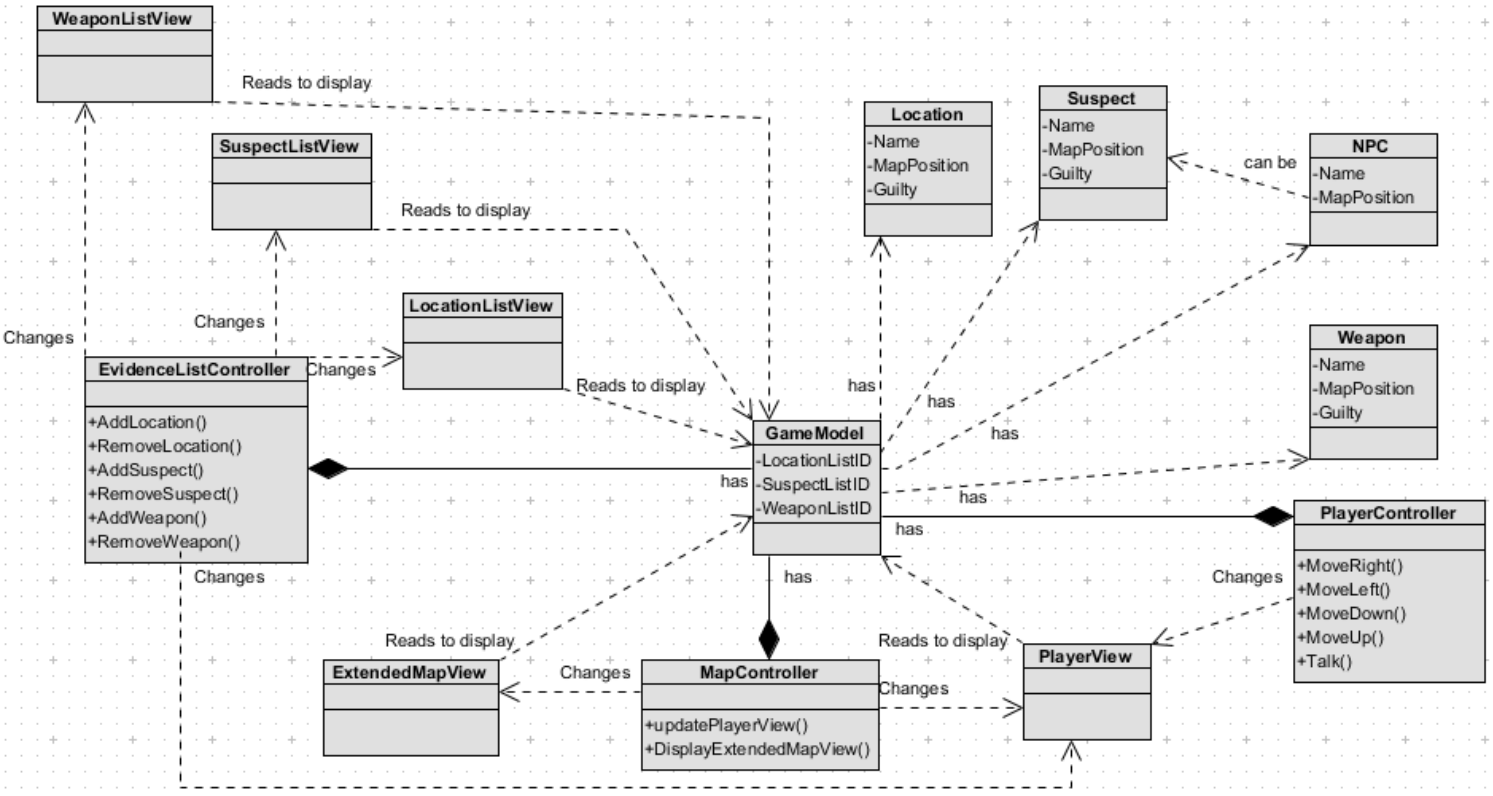


Screenshot 11

This is the segment of the map that the player is currently on. It is displayed on the main screen. The star is a pointer showing the players position on the map. *This is just an example.*



Class Diagram



Model

In my class diagram there is one model called Game Model. It contains all of the data for the game including the evidence lists. It also has locations, weapons, suspects and NPC's.

Controllers

There are three controllers. They are named; Player Controller, Map Controller and Evidence List Controller. Player Controller contains the methods; Move Right, Move Left, Move Down, Move Up and Talk which are all self-explanatory. Map Controller contains the methods Update Player View and Display Extended Map View which are also self-explanatory. Evidence List Controller contains the methods Add Location, Remove Location, Add Suspect, Remove Suspect, Add Weapon and Remove weapon which are also self-explanatory.

Views

There are five views. They are named; Player View, Extended Map View, Location List View, Suspect List View and Weapon List View. They all read the Game Model to display. Location List View, Suspect List View and Weapon List View are all changed by the Evidence List Controller. Extended Map View is changed by the Map Controller. Player view is changed by all three controllers.

Views

Suspect List

HTML

```
<!-- Suspect List -->
<div id="suspectList" style="position:absolute; top:0px; left:600px;">
    <canvas id="SuspectListCanvas" width="200" height="150" style="border:3px solid #000000;">
    </canvas>
```

I have put this canvas into a div element so that I could have more than one canvas on the page with the positioning that I choose.

JS

```
SuspectListView = Class( Object,
{
    ctx : (id("SuspectListCanvas")).getContext('2d'),
    title : '',
    items : [],
    setUp : function (pTitle,pItems)
    {
        this.title = pTitle;
        this.items = pItems;
    },
    draw : function ()
    {
        var top = 60;
        var heightStep = 20;
        var count = 0;

        this.ctx.font = "20px Georgia";
        this.ctx.fillText(this.title,30, 30);
        this.ctx.font = "15px Georgia";

        for ( role in this.items )
        {
            this.ctx.fillText(role + ' - '+this.items[role], 10,top + (heightStep*count));
                count++;
        }// for
    }// draw
}// Class
);
```

This is the code that Todd reworked for me so that it was in a class. This is defining it

```
var aSusList = new SuspectListView();
aSusList.setUp('Weapon List',
    {
        Wife : 'Emily Sims', Mistress : 'Celia Maldanado' , Cleaner: 'Hope Gilbert',
        Gardener :'Rodger Lee', Neighbour :'Jackie Wolfe'
    }
);
```

This is an instance of the class

```
aSusList.draw();
```

This instantiates the class

Weapon List

HTML

```
<!-- Weapon List -->
<div id="weaponList" style="position:absolute; top:153px; left:600px;">
   <canvas id="WeaponListCanvas" width="200" height="150" style="border:3px solid #000000;">
   </canvas>
</div>
```

I have put this canvas into a div element so that I could have more than one canvas on the page with the positioning that I choose.

JS

```
WeaponListView = Class( Object,
   {
      ctx : (id("WeaponListCanvas")).getContext('2d'),
      title : '',
      items : [],
      setUp : function (pTitle,pItems)
      {
         this.title = pTitle;
         this.items = pItems;
      },
      draw : function ()
      {
         var top = 60;
         var heightStep = 20;
         var count = 0;

         this.ctx.font = "20px Georgia";
         this.ctx.fillText(this.title,30, 30);
         this.ctx.font = "15px Georgia";

         for ( role in this.items )
         {
            this.ctx.fillText(role + ' - '+this.items[role], 10,top + (heightStep*count));
                  count++;
         }// for
      }// draw
   }// Class
);
```

I reused the code from the suspect list view and made changes to suit the weapon list view

```
var aWeapList = new WeaponListView();
aWeapList.setUp('Suspect List',
   {
      One : 'Knife', Two : 'Gun' , Three: 'Scarf', Four :'Cleaver', Five :'Poison'
   }
);
```

This is an instance of the class

```
aWeapList.draw();
```

This instantiates the class

Location List

HTML

```
<!-- Location List -->
<div id="locationList" style="position:absolute; top:306px; left:600px;">
   <canvas id="LocationListCanvas" width="200" height="150" style="border:3px solid #000000;">
   </canvas>
</div>
```

I have put this canvas into a div element so that I could have more than one canvas on the page with the positioning that I choose.

JS

```
LocationListView = Class( Object,
   {
      ctx : (id("LocationListCanvas")).getContext('2d'),
      title : '',
      items : [],
      setUp : function (pTitle,pItems)
      {
         this.title = pTitle;
         this.items = pItems;
      },
      draw : function ()
      {
         var top = 60;
         var heightStep = 20;
         var count = 0;

         this.ctx.font = "20px Georgia";
         this.ctx.fillText(this.title,30, 30);
         this.ctx.font = "15px Georgia";

         for ( role in this.items )
         {
            this.ctx.fillText(role + ' - '+this.items[role], 10,top + (heightStep*count));
                count++;
         }// for
      }// draw
   }// Class
);
```

I reused the code from the suspect list view and made changes to suit the location list view

```
var aLocList = new LocationListView();
aLocList.setUp('Location List',
   {
      One : 'Bedroom', Two : 'Bathroom' , Three: 'Lounge', Four :'Kitchen', Five :'Patio'
   }
);
```

This is an instance of the class

```
aLocList.draw();
```

This instantiates the class

Button Display

HTML

```
<!-- Buttons Display -->
<div id="buttonsDisplay"> <!-- style="position:absolute; top:460px; left:600px;" -->
   <p>
      <input type="button" id="mapButton"  value="Map" onClick='alert("there you have been alerted")'>
   </p>
   <p>
      <input type="button" id="solveButton" value="Solve" onClick='alert("there you have been alerted")'>
   </p>
</div>
```

Here I have not used a canvas due to having two buttons. I have put them both into paragraph tags inside a div element for positioning purposes

CSS

```
#buttonsDisplay {
    position:absolute;
    top:460px;
    left:600px;
    width:200px;
    height:137px;
    border:3px solid #000000;
}
```

This is for the display of the buttons

```
#mapButton, #solveButton {
  font-family: georgia;
  font-weight: bold;
  color: #000103 !important;
  font-size: 15px;
  text-shadow: 1px 1px 0px #7CACDE;
  box-shadow: 1px 1px 1px #BEE2F9;
  padding: 7px 22px;
  -moz-border-radius: 14px;
  -webkit-border-radius: 14px;
  border-radius: 14px;
  border: 4px outset #7F8087;
  background: #B5BBBD;
  position:relative;
  left:50px;
}
```

This sets the styling properties to both of the buttons

```
#mapButton:hover, #solveButton:hover {
  color: #000103 !important;
  background: #CCD0E6;
}
```

This is the code for the hover effect of the buttons

Player View (map)

HTML

```
<!-- Map Position -->
<div id="mapPosition" style="position:absolute; top:0px; left:0px;">
    <canvas id="MapPositionCanvas" width="596" height="456" style="border:3px solid #000000;">
    </canvas>
</div>
```

I have put this canvas into a div element so that I could have more than one canvas on the page with the positioning that I choose

JS

My code didn't want to work so for now I have done this;

```
function image()
{
    var c = document.getElementById("MapPositionCanvas");
    var ctx = c.getContext("2d");
    ctx.font = "30px Georgia";
    ctx.fillText("Map Image Will Go Here",60,100);
};

image();
```

I have put writing onto the canvas stating that an image will go here

Conclusion

I have started learning to code in JavaScript and have reused some of my knowledge of HTML and CSS in this stage of the project. I have found the MVC architecture a little confusing using notepad ++. I understand functions but making classes had me a little confused. With my research and Todd's help I have managed to get to the point that I am currently at.

Milestone Two

Introduction

Who Killed Mr Sims is a text based game that is based on solving a murder case. The purpose of this report is to show my understanding of the processes that I have undertaken in creating this game. As the project is indicative this is the second step of my work. This has been requested by Todd Cochrane and is due on Friday 17th October at 9 am. In the middle of the first half of the semester I came down with a really bad case of the flu. Because of this I had to take three weeks off where I was basically bed-ridden and couldn't study at all. This has affected my learning of the 'stuff' required for this milestone.

Findings

WampServer

For this milestone we had to download WampServer. We then had to do our assignment work on this server. When I save a file I save it in the www folder that is located in the wamp folder in my C drive. So now instead of clicking run in Notepad ++ I open a browser and type *localhost/* followed by the file name and extension that I want to open. To enable WampServer I had to go into my local services and disable MySQL as WampServer has its own MySQL incorporated into it. If I didn't do this WampServer wouldn't work.

WhoKilledMrSims – MySQL Database

In this stage of the project I was set the task of creating a database for my game. As my heading states the name of my database is WhokilledMrSims. WhoKilledMrSims contains seven tables; Suspect, NPC, Location, Weapon, MapPosition, Player and Game.

Suspect Table

This is what my Suspect table looks like in WampServers MySQL console;

```
mysql> SELECT * FROM Suspect;
+-----------+---------------------------+---------------+---------+---------+
| SuspectID | Name                      | MapPositionID | Display | Guilty  |
+-----------+---------------------------+---------------+---------+---------+
|         1 | Wife - Emily Sims         |             1 |       1 |       0 |
|         2 | Cleaner - Hope Gilbert    |             3 |       1 |       0 |
|         3 | Mistress - Celia Maldanado|             7 |       1 |       0 |
|         4 | Neighbour - Jackie Wolfe  |             9 |       1 |       0 |
|         5 | Gardener - Rodger Lee     |            10 |       1 |       0 |
+-----------+---------------------------+---------------+---------+---------+
5 rows in set (0.00 sec)
```

There are five suspects. They have a SuspectID, Name, MapPositionID, Display and Guilty. Display is for whether the suspect gets displayed on the screen. Guilty is for whether the suspect is the suspect of the game.

This is the create table statement;

```
CREATE TABLE Suspect
(
SuspectID INT NOT NULL AUTO_INCREMENT,
Name VARCHAR(50),
MapPositionID INT,
Guilty BOOLEAN,
PRIMARY KEY(SuspectID),
FOREIGN KEY(MapPositionID) REFERENCES MapPosition(MapPositionID)
);
```

As you can see it looks different from the actual table. I added a column called Display. It is a Boolean with a default of 1 which means true. I also added a default to guilty of 0 which means false. SuspectID is the primary key. MapPositionID is a foreign key that links to the MapPosition table.

NPC Table

This is what my NPC table looks like in WampServers MySQL console;

```
mysql> SELECT * FROM NPC;
+-------+--------------------------+---------------+
| NPCID | Name                     | MapPositionID |
+-------+--------------------------+---------------+
|     1 | Brother - Karl Sims      |             2 |
|     2 | Sister - Lydia Grant     |             4 |
|     3 | Son - Peter Sims         |             6 |
|     4 | Friend - Steve Mitchell  |             8 |
|     5 | Neighbour - Mark Wolfe   |             9 |
|     6 | Boss - Neil Hodges       |             7 |
+-------+--------------------------+---------------+
6 rows in set (0.00 sec)
```

NPCs are non-playing characters or in the case of this game non-suspects. There are six NPCs.

This is the create table statement;

```
CREATE TABLE NPC
(
NPCID INT NOT NULL AUTO_INCREMENT,
Name VARCHAR(50),
MapPositionID INT,
PRIMARY KEY(NPCID),
FOREIGN KEY(MapPositionID) REFERENCES MapPosition(MapPositionID)
);
```

NPCID is the primary key. MapPositionID id a foreign key that links to the MapPosition table.

Location Table

This is what my Location table looks like in WampServers MySQL console;

```
mysql> SELECT * FROM Location;
+------------+----------+---------------+---------+--------+
| LocationID | Name     | MapPositionID | Display | GUILTY |
+------------+----------+---------------+---------+--------+
|          1 | Bedroom  |             1 |       1 |      0 |
|          2 | Bathroom |             3 |       1 |      0 |
|          3 | Lounge   |             7 |       1 |      0 |
|          4 | Kitchen  |             9 |       1 |      0 |
|          5 | Patio    |            10 |       1 |      0 |
+------------+----------+---------------+---------+--------+
5 rows in set (0.00 sec)
```

Locations are the possible spots of the murder. There are five locations. They have a LocationID, Name, MapPositionID, Display and Guilty. Display is for whether the suspect gets displayed on the screen. Guilty is for whether the suspect is the suspect of the game.

This is the create table statement;

```
CREATE TABLE Location
(
LocationID INT NOT NULL AUTO_INCREMENT,
Name VARCHAR(50),
MapPositionID INT,
Guilty BOOLEAN,
PRIMARY KEY(LocationID),
FOREIGN KEY(MapPositionID) REFERENCES MapPosition(MapPositionID)
);
```

As you can see it looks different from the actual table. I added a column called Display. It is a Boolean with a default of 1 which means true. I also added a default to guilty of 0 which means false. LocationID is the primary key. MapPositionID is a foreign key that links to the MapPosition table.

Weapon Table

This is what my Weapon table looks like in WampServers MySQL console;

```
mysql> SELECT * FROM Weapon;
+----------+---------+---------------+---------+--------+
| WeaponID | Name    | MapPositionID | Display | Guilty |
+----------+---------+---------------+---------+--------+
|        1 | Scarf   |             1 |       1 |      0 |
|        2 | Poison  |             3 |       1 |      0 |
|        3 | Knife   |             7 |       1 |      0 |
|        4 | Cleaver |             9 |       1 |      0 |
|        5 | Gun     |            10 |       1 |      0 |
+----------+---------+---------------+---------+--------+
5 rows in set (0.00 sec)
```

There are five weapons. They have a WeaponID, Name, MapPositionID, Display and Guilty. Display is for whether the suspect gets displayed on the screen. Guilty is for whether the suspect is the suspect of the game.

This is the create table statement;

```
CREATE TABLE Weapon
(
WeaponID INT NOT NULL AUTO_INCREMENT,
Name VARCHAR(50),
MapPositionID INT,
Guilty BOOLEAN,
PRIMARY KEY(WeaponID),
FOREIGN KEY(MapPositionID) REFERENCES MapPosition(MapPositionID)
);
```

As you can see it looks different from the actual table. I added a column called Display. It is a Boolean with a default of 1 which means true. I also added a default to guilty of 0 which means false. WeaponID is the primary key. MapPositionID is a foreign key that links to the MapPosition table.

MapPosition Table

This is what my MapPosition table looks like in WampServers MySQL console;

```
mysql> SELECT * FROM MapPosition;
+---------------+-----------+-------------+
| MapPositionID | RoomName  | MapLocation |
+---------------+-----------+-------------+
|             1 | Bedroom   | 1           |
|             2 | Hallway1  | 2           |
|             3 | Bathroom  | 3           |
|             4 | Hallway2  | 4           |
|             5 | Crossroad | 5           |
|             6 | Hallway3  | 6           |
|             7 | Lounge    | 7           |
|             8 | Hallway4  | 8           |
|             9 | Kitchen   | 9           |
|            10 | Patio     | 10          |
+---------------+-----------+-------------+
10 rows in set (0.00 sec)
```

There are 10 MapPositions that correspond to the map of the game.

This is the create table statement;

```
CREATE TABLE MapPosition
(
MapPositionID INT NOT NULL AUTO_INCREMENT,
RoomName VARCHAR(50),
MapLocation VARCHAR(50),
PRIMARY KEY(MapPositionID)
);
```

MapPositionID is the primary key.

Player Table

This is what my Player table looks like in WampServers MySQL console;

```
mysql> SELECT * FROM Player;
Empty set (0.00 sec)
```

This table is empty as no players have played a game yet.

This is the create table statement;

```
CREATE TABLE Player
(
PlayerID INT NOT NULL AUTO_INCREMENT,
MapPositionID INT,
Solved BOOLEAN,
PRIMARY KEY(PlayerID),
FOREIGN KEY(MapPositionID) REFERENCES MapPosition(MapPositionID)
);
```

There are three columns. A player will have a PlayerID, MapPositionID and Solved. Solved is whether their guess was correct or not. PlayerID is the primary key. MapPositionID is a foreign key linking to the MapPosition table. This points to where the player is currently positioned.

Game Table

This is what my Player table looks like in WampServers MySQL console;

```
mysql> SELECT * FROM Game;
+--------+-----------+----------+------------+
| GameID | SuspectID | WeaponID | LocationID |
+--------+-----------+----------+------------+
|      1 |         1 |        1 |          1 |
|      2 |         1 |        1 |          2 |
|      3 |         1 |        1 |          3 |
|      4 |         1 |        1 |          4 |
|      5 |         1 |        1 |          5 |
|      6 |         1 |        2 |          1 |
|      7 |         1 |        2 |          2 |
|      8 |         1 |        2 |          3 |
|      9 |         1 |        2 |          4 |
|     10 |         1 |        2 |          5 |
|     11 |         1 |        3 |          1 |
```

I have only included a snapshot of part of the table as there are 125 games. Each game has a suspect, weapon and location. These are the correct answers for the particular game.

This is the create table statement;

```
CREATE TABLE Game
(
GameID INT NOT NULL AUTO_INCREMENT,
SuspectID INT,
WeaponID INT,
LocationID INT,
PRIMARY KEY(GameID),
FOREIGN KEY(SuspectID) REFERENCES Suspect(SuspectID),
FOREIGN KEY(WeaponID) REFERENCES Weapon(WeaponID),
FOREIGN KEY(LocationID) REFERENCES Location(LocationID)
);
```

GameID is the primary key. There are three foreign keys that link to their respective tables. They are: SuspectID, WeaponID and LocationID.

Linking to my Database

To link my database to my game I am supposed to use PHP. I have the code that Todd provided the class. It is in a file called databaseJSON.php.

First you have to call the PHP file in JavaScript.

```
//call the php file and JSONs the result assigning it to the view
    oReq = new XMLHttpRequest();
        function setcanvas(pTxt)
         {
            var aCanvas = document.getElementById("SuspectListCanvas");
            aCanvas.innerHTML = pTxt;
         }
         function handler()
         {
            if(oReq.readyState == 4)
            {
                if(oReq.status == 200)
                {
                 //alert("got a response");
                 //setDiv(oReq.responseText);
                 jVar = JSON.parse(oReq.responseText);
                 strThe = '';

                 for (var rec in jVar)
                      strThe     += jVar[rec].Name;
                  aSuspectListView = new SuspectListView();
                  aSuspectListView.draw(strThe);

                // objGameReplyView = new gameReplyView();
                // objGameReplyView.draw(strThe);

                }
            }
```

```
function send()
    {

        if(oReq)
        {

            oReq.open("GET","databaseJSON.php",true);
            oReq.onreadystatechange = handler;
            oReq.send();
        }
        else
        alert( 'No oReq');
    }
```

Then the PHP file connects to the database.

```
function connect()
{
   // Connect to the database
   //   Check for success
   try {
      $this->mysqli = new mysqli("localhost", "root", "", "WhoKilledMrSims");
   }
   catch (mysqli_sql_exception $e) {
      throw $e;
   }
   /*
   if ($this->mysqli->connect_errno) {
      exit("Failed to connect to MySQL: (" . $this->mysqli->connect_errno . ") " . $this->mysqli->connect_error);
   }
   */
}
```

(this code is inside a PHP file)

Displaying data

To display data you must first have a query. The following code will run a query using a variable as a placeholder.

```
function query($pSQL)
{
   // Run Query
   try {
      $this->res = $this->mysqli->query($pSQL);
   }
   catch (mysqli_sql_exception $e) {
      throw $e;
   }
   /*
   if( !($this->res = $this->mysqli->query($pSQL))){
   //    Check for success
     exit("Query to MySQL: $SQL error : ". $this->mysqli->error);
   }
   */
}
```

This code will display the results of the query.

```
function display($pName)
{
    $allRows = Array();
   $rowCount = 0;
    // produce results

    while ($row = $this->res->fetch_assoc()) {
        $allRows[$rowCount] = $row;
        $rowCount++;
    }
    echo json_encode(new ArrayValue($allRows),JSON_PRETTY_PRINT);
}
```

This code is the creation of the actual query, running of the query and displaying it.

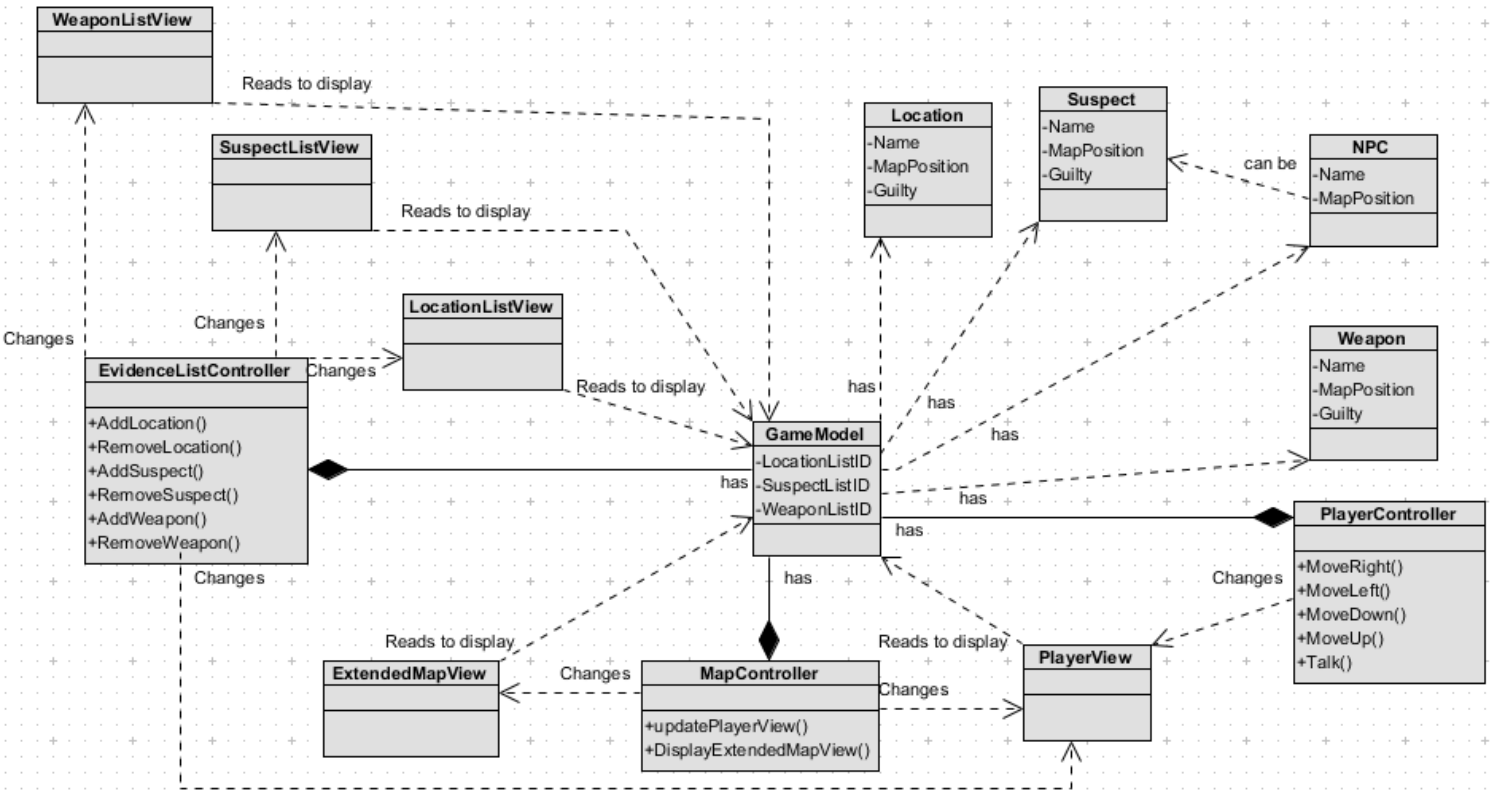
```
try
{
    $aDB = new DaBase();

    $SQL = "SELECT Name FROM Suspect";
    // $SQL = "SELECT Name FROM Suspect WHERE Guilty = 0";


    $aDB->query($SQL);
    $aDB->display("Suspects");
    $aDB->closeDb();
}
catch ( Exception $e)
{
    echo $e->getMessage();
}
```

My code from databaseJSON.php was given to me from a classmate that I changed to suit my needs. However it does not seem to work for me.

Class Diagram



Model

In my class diagram there is one model called Game Model. It contains all of the data for the game including the evidence lists. It also has locations, weapons, suspects and NPC's.

Controllers

There are three controllers. They are named; Player Controller, Map Controller and Evidence List Controller. Player Controller contains the methods; Move Right, Move Left, Move Down, Move Up and Talk which are all self-explanatory. Map Controller contains the methods Update Player View and Display Extended Map View which are also self-explanatory. Evidence List Controller contains the methods Add Location, Remove Location, Add Suspect, Remove Suspect, Add Weapon and Remove weapon which are also self-explanatory.

Views

There are five views. They are named; Player View, Extended Map View, Location List View, Suspect List View and Weapon List View. They all read the Game Model to display. Location List View, Suspect List View and Weapon List View are all changed by the Evidence List Controller. Extended Map View is changed by the Map Controller. Player view is changed by all three controllers.

Conclusion

I was really stoked with my marks for the first milestone due to not being sure about it. I have had a bit of struggle with this milestone due to being sick.

Milestone Three

Introduction

Who Killed Mr Sims is a text based game that is based on solving a murder case. The purpose of this report is to show my understanding on the processes that I have undertaken in creating this game. As the project in indicative this is the third and final step of my work. This has been requested by Todd Cochrane and is due on Friday 21st November at 9 am. I however have and extension so it is due on Friday 28th November at 9 am.

In the middle of the first half of the semester I came down with a really bad case of the flu. Because of this I had to take three weeks off where I was basically bed-ridden and couldn't study at all. This has affected my learning of the 'stuff' required for this milestone. Meaning that everything in this milestone is guess work. Hopefully it will be enough.

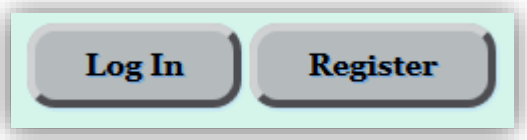
Findings

Login

This is the html code for the buttons

```
<!-- Log In -->
<div id="logIn" style="position:absolute; top:0px; left:800px; <!--border:3px solid #000000;-->">
    <input type="button" id="loginButton"  value="Log In" onClick='alert("there you have been alerted")'> <!-- Log In Button -->
    <input type="button" id="registerButton" value="Register" onClick='alert("there you have been alerted")'> <!-- Register Button -->
</div>
```

This is what it looks like in a browser



To facilitate the use of a login I added a table to my database. It is called User.

```
mysql> CREATE TABLE User
    -> (
    -> PlayerID INT NOT NULL AUTO_INCREMENT,
    -> Username VARCHAR(40) NOT NULL,
    -> Password VARCHAR(20) NOT NULL DEFAULT 'P@ssword1',
    -> Email VARCHAR(60),
    -> Status VARCHAR(20) NOT NULL DEFAULT 'Offline',
    -> Wins INT NOT NULL DEFAULT 0,
    -> Losses INT NOT NULL DEFAULT 0,
    -> PRIMARY KEY(PlayerID)
    -> );
Query OK, 0 rows affected (0.39 sec)
```

```
mysql> SELECT * FROM User;
+----------+----------+-----------+--------------------+---------+------+--------+
| PlayerID | Username | Password  | Email              | Status  | Wins | Losses |
+----------+----------+-----------+--------------------+---------+------+--------+
|        1 | Jnamu    | P@ssword1 | beka@hotmail.com   | Offline |    0 |      0 |
|        2 | Bubblez  | Jk2k4K    | Julie@hotmail.com  | Offline |    0 |      0 |
+----------+----------+-----------+--------------------+---------+------+--------+
2 rows in set (0.00 sec)
```

Login Tutorials

First try

I followed this tutorial. I created the required database and php files. It didn't work. You can find the files in the folder named login-test.
('PHP Login script tutorial', n.d.)

Second try

I then found this tutorial. I again created the required database and files. I fixed the problems that the html file had. I entered a user but it came back with an alert saying it was incorrect and then a list of errors. Due to time I am not able to try fix it. I have mentioned this to show that I have been trying to do the work even though I am really short on time. You can files the files in the folder named login-test-2.
('How to Create a Login System in PHP & MySQLi?', n.d.)

Interaction

I have not done any work for this as I have officially run out of time. I have however though a little bit about how I would achieve multiplayer interaction. I came to the conclusion that I would have attempted to create a chat bar. If I was to properly do this game and deploy it, I would give the option of proper multiplayer gameplay. There would be the option of working as a team or working against each other.

Conclusion

Due to being ill I have not accomplished much in this milestone as I had a very short amount of time to do it. I am not using this as an excuse, just stating the truth. Because of this, I have added things into this milestone that I have done to try and show that I have made an effort. I have struggled a lot and even other classmates have had a lot of issues, even when they were there for most classes.

Conclusion

This has been a very tricky class for me overall. I am new to JavaScript, PHP and the html canvas element. Once I got used to using the canvas element we went onto JavaScript. I thought I was doing okay until PHP was introduced and then I just got completely confused. Some aspects I can

understand and I understand most of the theory of it but the code just seems more and more confusing every time I look at it.

Bibliography

How to Create a Login System in PHP & MySQLi? (n.d.). Retrieved 26 November 2014, from

http://www.onlinetuting.com/create-login-script-in-php/

PHP Login script tutorial. (n.d.). Retrieved 26 November 2014, from

http://www.phpeasystep.com/phptu/6.html