```
1  using System;
2  using Microsoft.Xna.Framework;
3  using Microsoft.Xna.Framework.Graphics;
4  using Microsoft.Xna.Framework.Input;
5  using System.Data.SQLite;
6  using System.Collections.Generic;
7  using System.Threading;
8
9  namespace coursework
10 {
11     public class Game1 : Game
12     {
13         //initialise thread to manage database inputs
14         public static Thread DB = new Thread(DBManager.commandQueue);
15         public static bool GameActive;
16
17         private GraphicsDeviceManager graphics;
18         private SpriteBatch spriteBatch;
19         Graph graph;
20         #region buttons
21         //menu
22         Button Quit;
23         Button NewCharacter;
24         Button Menu;
25         Button Map;
26         Button Map2;
27         Button LoadCharacter;
28         //Character Builder
29         Textbox Name;
30         Textbox Username;
31         Button Elf;
32         Button Human;
33         Button Dwarf;
34         Button LevelUp;
35         Button LevelDown;
36         Textbox Dex;
37         Textbox Strength;
38         Button Dice;
39         Button Save;
40         //Load
41         Button enter;
42         Button nextPage;
43         Button lastPage;
44         Textbox ID;
45         Button Load;
46         //map
47         Button attack;
48         Button dash;
49         Button dodge;
50         Button endTurn;
51         //fight
52         Button d20;
53         Button hitDie;
54         Button ReturnToMap;
55         #endregion
56
```

```
57          Character character;
58          Weapon weapon;
59          Enemy enemy;
60          Weapon enemyweapon;
61          bool damageDone = false;
62          bool moved = false;
63          bool actionTaken = false;
64          bool dodged = false;
65          bool saveSuccessful = true;
66          //enemy attack
67          bool attackdieRolled = false;
68          bool HitDieRolled = false;
69          int roll1 = 0;
70          int roll2 = 0;
71          int attackroll = 0;
72          int playerDamage = 0;
73
74          #region textures
75          private Texture2D concrete;
76          private Texture2D grass;
77          private Texture2D wall;
78          private Texture2D tree;
79          private Texture2D sand;
80          private Texture2D water;
81          private Texture2D button;
82          private Texture2D redSquare;
83          private Texture2D blueSquare;
84          private Texture2D greenSquare;
85          private Texture2D die;
86          private Texture2D d20Image;
87          private SpriteFont font;
88          private SpriteFont smallFont;
89          private Texture2D CharacterIcon;
90          private Texture2D awakenedShrub;
91          #endregion
92          public string Race = "";
93          int dexMod;
94          int strMod;
95          string prevRace;
96          int speed;
97          public int Level = 1;
98          public int dex = 0;
99          public int str = 0;
100         public int AC = 10;
101         int playerX, playerY;
102         bool validUser = false;
103         bool playersturn = true;
104         int[] skillModifiers = new int[] { -5, -4, -3, -2, -1, 0, 1, 2, 3, 4,
                5, 6, 7, 8, 9, 10 };
105         Random rand = new Random();
106         public static int pageNum = 0;
107
108         enum GameState
109         {
110             Menu,
111             CharacterMaker,
```

```
112             LoadCharacter,
113             LoadWeapon,
114             Map,
115             Fight,
116             GameOver
117         }
118         private GameState gameState;
119 
120         #region maps
121 
122         //map with Lshaped wall in center
123         static int[,] map1 = new int[,]
124         {
125             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
126             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
127             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
128             { 0, 0, 0, 0, 0, 0, 5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0 },
129             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0 },
130             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0 },
131             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0 },
132             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0 },
133             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0 },
134             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
135             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
136             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
137 
138         };
139 
140         //map using all textures
141         static int[,] map2 = new int[,]
142         {
143             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 4, 1 },
144             { 0, 0, 0, 0, 5, 5, 5, 5, 5, 1, 1, 2, 2, 2, 3, 3, 3, 3, 2, 1 },
145             { 0, 0, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 3, 2, 1 },
146             { 0, 0, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 3, 2, 1 },
147             { 0, 0, 5, 1, 1, 3, 3, 3, 3, 1, 1, 1, 2, 2, 2, 3, 3, 2, 1, 1 },
148             { 0, 0, 5, 1, 1, 3, 3, 3, 1, 1, 4, 4, 1, 2, 2, 2, 2, 2, 1, 1 },
149             { 0, 0, 5, 1, 1, 3, 3, 1, 1, 1, 4, 1, 1, 1, 2, 2, 2, 0, 0, 0 },
150             { 0, 0, 5, 4, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 4, 0, 0, 0, 0, 0 },
151             { 0, 0, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1 },
152             { 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 4, 1, 1 },
153             { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 4, 1, 1 },
154             { 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 4, 1, 1 }
155         };
156 
157         //array to show coordinates of square the player can move to
158         //0 = out of range, 1 = in range
159         int[,] move = new int[,]
160         {
161             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
162             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
163             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
164             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
165             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
166             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
167             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
```

```
168             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
169             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
170             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
171             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
172             { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
173
174         };
175
176         static int[,] map;
177         int PlayerLocation;
178         int EnemyLocation;
179
180         #endregion
181
182         public Game1()
183         {
184             graphics = new GraphicsDeviceManager(this);
185             Content.RootDirectory = "Content";
186             IsMouseVisible = true;
187         }
188
189         protected override void Initialize()
190         {
191             //example character, weapon and enemy to play the game with
192             character = new Character("Anya", "Elf", 1, 15, 7, 12, 30, 20,
                  true);
193             weapon = new Weapon("spear", 20, 6, true, false);
194             enemy = new Enemy("awakened shrub", 8, 3, 9, 25, 10, 4, 9,
                  false);
195             enemyweapon = new Weapon("", 0, 0, false, false);
196
197             //start the database thread
198             GameActive = true;
199             DB.Start();
200
201
202             IsMouseVisible = true;
203             base.Initialize();
204         }
205
206         protected override void LoadContent()
207         {
208             spriteBatch = new SpriteBatch(GraphicsDevice);
209
210
211             //load in textures/font
212             #region Textures
213             font = Content.Load<SpriteFont>("font");
214             smallFont = Content.Load<SpriteFont>("smallFont");
215             concrete = Content.Load<Texture2D>("concrete");
216             wall = Content.Load<Texture2D>("wall");
217             grass = Content.Load<Texture2D>("grass");
218             tree = Content.Load<Texture2D>("tree");
219             sand = Content.Load<Texture2D>("sand");
220             water = Content.Load<Texture2D>("water");
221             button = Content.Load<Texture2D>("SimpleButton");
```

```
222             redSquare = Content.Load<Texture2D>("redSquare");
223             greenSquare = Content.Load<Texture2D>("greenSquare");
224             blueSquare = Content.Load<Texture2D>("blueSquare");
225             die = Content.Load<Texture2D>("dice");
226             d20Image = Content.Load<Texture2D>("d20");
227             awakenedShrub = Content.Load<Texture2D>("awakenedShrub");
228             CharacterIcon = Content.Load<Texture2D>("character");
229             #endregion
230
231             //initialise buttons and assign click methods
232             #region Buttons
233             //menu
234             Quit = new Button(new Rectangle(100, 300, 200, 75), button, font,
                    "QUIT");
235             Quit.Click += Quit_Click;
236
237             NewCharacter = new Button(new Rectangle(100, 100, 300, 75),
                  button, font, "New Character");
238             NewCharacter.Click += NewCharacter_Click;
239
240             LoadCharacter = new Button(new Rectangle(100, 200, 300, 75),
                  button, font, "Load Character");
241             LoadCharacter.Click += LoadCharacter_Click;
242
243             Map = new Button(new Rectangle(560, 50, 100, 50), button, font,
                  "Map");
244             Map.Click += Map_Click;
245
246             Map2 = new Button(new Rectangle(560, 150, 100, 50), button, font,
                    "Map 2");
247             Map2.Click += Map2_Click;
248
249             //Character Builder
250             Name = new Textbox(new Rectangle(20, 150, 220, 40), button, font,
                    "", "str", 15);
251
252             Username = new Textbox(new Rectangle(300, 150, 220, 40), button,
                  font, "", "str", 15);
253
254             Elf = new Button(new Rectangle(20, 335, 120, 50), button, font,
                  "Elf");
255             Elf.Click += Elf_click;
256
257             Human = new Button(new Rectangle(20, 280, 120, 50), button, font,
                    "Human");
258             Human.Click += Human_click;
259
260             Dwarf = new Button(new Rectangle(20, 390, 120, 50), button, font,
                    "Dwarf");
261             Dwarf.Click += Dwarf_click;
262
263             LevelUp = new Button(new Rectangle(270, 280, 40, 40), button,
                  font, "+");
264             LevelUp.Click += LevelUp_click;
265
266             LevelDown = new Button(new Rectangle(170, 280, 40, 40), button,
```

```
                font, "-");
267             LevelDown.Click += LevelDown_click;
268
269             Dex = new Textbox(new Rectangle(410, 280, 40, 40), button, font,
                  "0", "int", 2);
270
271             Strength = new Textbox(new Rectangle(410, 335, 40, 40), button,
                  font, "0", "int", 2);
272
273             Dice = new Button(new Rectangle(340, 385, 40, 34), die, font,
                  "");
274             Dice.Click += Dice_click;
275
276             Save = new Button(new Rectangle(560, 400, 100, 50), button, font,
                   "Save");
277             Save.Click += Save_click;
278
279             Menu = new Button(new Rectangle(690, 10, 100, 40), button, font,
                  "Menu");
280             Menu.Click += Menu_Click;
281
282             //Load
283             enter = new Button(new Rectangle(540, 200, 70, 40), button, font,
                   "Enter");
284             enter.Click += enter_click;
285
286             nextPage = new Button(new Rectangle(640, 420, 70, 40), button,
                  font, "Next");
287             nextPage.Click += nextPage_click;
288
289             lastPage = new Button(new Rectangle(40, 420, 70, 40), button,
                  font, "Back");
290             lastPage.Click += lastPage_click;
291
292             ID = new Textbox(new Rectangle(300, 420, 50, 40), button, font,
                  "", "int", 3);
293
294             Load = new Button(new Rectangle(360, 420, 70, 40), button, font,
                  "Load");
295             Load.Click += Load_click;
296
297             //fight
298             attack = new Button(new Rectangle(345, 430, 70, 40), button,
                  smallFont, "Attack");
299             attack.Click += attack_click;
300
301             dash = new Button(new Rectangle(420, 430, 60, 40), button,
                  smallFont, "Dash");
302             dash.Click += dash_click;
303
304             dodge = new Button(new Rectangle(485, 430, 60, 40), button,
                  smallFont, "Dodge");
305             dodge.Click += dodge_click;
306
307             endTurn = new Button(new Rectangle(550, 430, 80, 40), button,
                  smallFont, "endTurn");
```

```
308             endTurn.Click += endTurn_click;
309
310             d20 = new Button(new Rectangle(400, 120, 80, 80), d20Image, font,
                    "");
311             d20.Click += d20_click;
312
313             hitDie = new Button(new Rectangle(400, 280, 80, 80), d20Image,
                  font, "");
314             hitDie.Click += hitDie_click;
315
316             ReturnToMap = new Button(new Rectangle(300, 430, 200, 40),
                  button, font, "Return To Map");
317             ReturnToMap.Click += ReturnToMap_click;
318             #endregion
319         }
320
321         #region Click Methods
322         /// <summary>
323         /// returns player to the map screen
324         /// damageDone flag set to false
325         /// </summary>
326         /// <param name="sender"></param>
327         /// <param name="e"></param>
328         private void ReturnToMap_click(object sender, EventArgs e)
329         {
330             gameState = GameState.Map;
331             damageDone = false;
332             if(!playersturn)
333             {
334                 playersturn = true;
335                 dodged = false;
336                 attackdieRolled = false;
337                 HitDieRolled = false;
338             }
339         }
340
341         /// <summary>
342         /// roll hit die
343         /// </summary>
344         /// <param name="sender"></param>
345         /// <param name="e"></param>
346         private void hitDie_click(object sender, EventArgs e)
347         {
348             hitDie.text = rand.Next(1, weapon.hitDie+1).ToString();
349         }
350
351         /// <summary>
352         /// roll d20
353         /// </summary>
354         /// <param name="sender"></param>
355         /// <param name="e"></param>
356         private void d20_click(object sender, EventArgs e)
357         {
358             d20.text = rand.Next(1, 21).ToString();
359         }
360
```

```
361         /// <summary>
362         /// end the players turn
363         /// </summary>
364         /// <param name="sender"></param>
365         /// <param name="e"></param>
366         private void endTurn_click(object sender, EventArgs e)
367         {
368             playersturn = false;
369         }
370
371
372         private void attack_click(object sender, EventArgs e)
373         {
374             if(!actionTaken)
375             {
376                 gameState = GameState.Fight;
377                 d20.text = "";
378                 hitDie.text = "";
379                 actionTaken = true;
380             }
381         }
382         private void dash_click(object sender, EventArgs e)
383         {
384             if (!actionTaken && moved)
385             {
386                 moved = false;
387                 actionTaken = true;
388             }
389         }
390         private void dodge_click(object sender, EventArgs e)
391         {
392             if (!actionTaken)
393             {
394                 dodged = true;
395                 actionTaken = true;
396             }
397         }
398         /// <summary>
399         /// load character info from characters and race tables to build
              charcter object for use in fight
400         /// </summary>
401         /// <param name="sender"></param>
402         /// <param name="e"></param>
403         private void Load_click(object sender, EventArgs e)
404         {
405             if (gameState == GameState.LoadCharacter)
406             {
407                 SQLiteConnection conn;
408                 conn = DBManager.CreateConnection();
409                 SQLiteDataReader sqlite_datareader;
410                 SQLiteCommand sqlite_cmd;
411                 sqlite_cmd = conn.CreateCommand();
412                 sqlite_cmd.CommandText = "SELECT * FROM Characters, Race
                    WHERE Characters.CharacterID = '" + ID.text + "'
                    Characters.Race = Race.RaceName";
413                 sqlite_datareader = sqlite_cmd.ExecuteReader();
```

```
414                 string myreader0 = sqlite_datareader.GetInt32(0).ToString();
415                 string myreader1 = sqlite_datareader.GetString(1);
416                 string myreader2 = sqlite_datareader.GetInt32(2).ToString();
417                 string myreader3 = sqlite_datareader.GetString(3);
418                 string myreader4 = sqlite_datareader.GetInt32(4).ToString();
419                 string myreader5 = sqlite_datareader.GetInt32(5).ToString();
420                 string myreader6 = sqlite_datareader.GetInt32(6).ToString();
421                 //character = new Character(myreader0,my);
422                 //conn.Close();
423             }
424         }
425 
426         private void Dice_click(object sender, EventArgs e)
427         {
428             dex = rand.Next(1, 21);
429             Dex.text = dex.ToString();
430             str = rand.Next(1, 21);
431             Strength.text = str.ToString();
432         }
433 
434         private void enter_click(object sender, EventArgs e)
435         {
436             if (Username.text != "")
437             {
438                 validUser = true;
439             }
440         }
441 
442         private void nextPage_click(object sender, EventArgs e)
443         {
444             pageNum++;
445         }
446 
447         private void lastPage_click(object sender, EventArgs e)
448         {
449             pageNum--;
450         }
451 
452         private void Map_Click(object sender, EventArgs e)
453         {
454             gameState = GameState.Map;
455             map = map1;
456             PlayerLocation = (map.GetLength(0) - 1) * map.GetLength(1);  //  ↪
                <-- player start in bottom left corner
457             EnemyLocation = map.GetLength(1) - 1;  //<-- enemy start in top  ↪
                right corner
458         }
459         private void Map2_Click(object sender, EventArgs e)
460         {
461             gameState = GameState.Map;
462             map = map2;
463             PlayerLocation = (map.GetLength(0) - 1) * map.GetLength(1);  //  ↪
                <-- player start in bottom left corner
464             EnemyLocation = map.GetLength(1) - 1;  //<-- enemy start in top  ↪
                right corner
465         }
```

```
466
467         private void Menu_Click(object sender, EventArgs e)
468         {
469             gameState = GameState.Menu;
470             validUser = false;
471         }
472
473         private void LoadCharacter_Click(object sender, EventArgs e)
474         {
475             gameState = GameState.LoadCharacter;
476         }
477
478         // change so can't save if username is empty
479         private void Save_click(object sender, EventArgs e)
480         {
481             if (Username.text != "")
482             {
483                 if (DBManager.commands.Count == 0)
484                 {
485                     DB.Interrupt();
486                 }
487                 DBManager.InsertData(Name.text, Race, Level, (int.Parse
                      (Dex.text) + dexMod).ToString(), (int.Parse(Strength.text)
                      + strMod).ToString(), AC, Username.text);
488                 saveSuccessful = true;
489
490             }
491             else
492             {
493                 saveSuccessful = false;
494             }
495         }
496
497         private void LevelDown_click(object sender, EventArgs e)
498         {
499             if (Level > 1)
500             { Level--; }
501         }
502
503         private void LevelUp_click(object sender, EventArgs e)
504         {
505             if (Level < 20)
506             { Level++; }
507         }
508
509         private void Human_click(object sender, EventArgs e)
510         {
511             Race = "Human";
512         }
513
514         private void Elf_click(object sender, EventArgs e)
515         {
516             Race = "Elf";
517         }
518
519         private void Dwarf_click(object sender, EventArgs e)
```

```
520         {
521             Race = "Dwarf";
522         }
523
524         private void NewCharacter_Click(object sender, EventArgs e)
525         {
526             gameState = GameState.CharacterMaker;
527         }
528
529         private void Quit_Click(object sender, System.EventArgs e)
530         {
531             GameActive = false;
532             Exit();
533         }
534         #endregion
535
536         protected override void Update(GameTime gameTime)
537         {
538             if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
                  ButtonState.Pressed || Keyboard.GetState().IsKeyDown
                  (Keys.Escape))
539                 Exit();
540
541             // TODO: Add your update logic here
542
543             base.Update(gameTime);
544         }
545
546         protected override void Draw(GameTime gameTime)
547         {
548
549             GraphicsDevice.Clear(Color.FloralWhite);
550             // TODO: Add your drawing code here
551
552             MouseState mouseState = Mouse.GetState();
553             spriteBatch.Begin();
554             switch (gameState)
555             {
556
557                 case GameState.Menu:
558                     GraphicsDevice.Clear(Color.DarkRed);
559                     #region menu
560                     spriteBatch.DrawString(font, "Medieval English", new
                    Vector2(100, 100), Color.Pink);
561                     Quit.Update(mouseState, spriteBatch);
562                     NewCharacter.Update(mouseState, spriteBatch);
563                     LoadCharacter.Update(mouseState, spriteBatch);
564                     Map.Update(mouseState, spriteBatch);
565                     Map2.Update(mouseState, spriteBatch);
566                     #endregion
567                     break;
568                 case GameState.CharacterMaker:
569                     #region CharacterMaker
570                     spriteBatch.DrawString(font, "Character Builder", new
                    Vector2(280, 10), Color.Black);
571                     spriteBatch.DrawString(font, "Name: " + Name.text, new
```

```
                    Vector2(560, 55), Color.Black);
572                 spriteBatch.DrawString(font, "Race: " + Race, new Vector2
                    (560, 90), Color.Black);
573                 spriteBatch.DrawString(font, "Level: " + Level, new
                    Vector2(560, 125), Color.Black);
574
575                 spriteBatch.DrawString(font, "Race", new Vector2(40,
                    230), Color.Black);
576                 Human.Update(mouseState, spriteBatch);
577                 Elf.Update(mouseState, spriteBatch);
578                 Dwarf.Update(mouseState, spriteBatch);
579
580                 spriteBatch.DrawString(font, "Level", new Vector2(210,
                    230), Color.Black);
581                 LevelUp.Update(mouseState, spriteBatch);
582                 spriteBatch.DrawString(font, Level.ToString(), new
                    Vector2(235, 280), Color.Black);
583                 LevelDown.Update(mouseState, spriteBatch);
584
585                 if(saveSuccessful == false)
586                 {
587                     spriteBatch.DrawString(smallFont, "UserName
                    Required", new Vector2(580, 350), Color.DarkRed);
588                 }
589                 Save.Update(mouseState, spriteBatch);
590                 Menu.Update(mouseState, spriteBatch);
591
592                 spriteBatch.DrawString(font, "Name", new Vector2(40,
                    110), Color.Black);
593                 Name.Update(mouseState, spriteBatch);
594
595                 spriteBatch.DrawString(font, "UserName", new Vector2(320,
                     110), Color.Black);
596                 Username.Update(mouseState, spriteBatch);
597
598                 spriteBatch.DrawString(font, "Stats", new Vector2(360,
                    230), Color.Black);
599                 spriteBatch.DrawString(font, "Dex", new Vector2(340,
                    280), Color.Black);
600                 Dex.Update(mouseState, spriteBatch);
601                 spriteBatch.DrawString(font, "Str", new Vector2(335,
                    335), Color.Black);
602                 Strength.Update(mouseState, spriteBatch);
603
604                 Dice.Update(mouseState, spriteBatch);
605
606                 if (Race != prevRace)
607                 {
608                     SQLiteConnection sqlite_conn;
609                     sqlite_conn = DBManager.CreateConnection(); //cant
                    read and write at the same time, trying to read constantly
                     for modifiers is locking it
610                     if (Race != "")
611                     {
612                         DBManager.ReadModifiers(sqlite_conn, ref dexMod,
                    ref strMod, Race, ref speed);
```

```
613                         }
614                         sqlite_conn.Close();
615                         prevRace = Race;
616                     }
617                     spriteBatch.DrawString(font, "Race", new Vector2(465,
                    200), Color.Black);
618                     spriteBatch.DrawString(font, "Modifiers", new Vector2
                    (450, 230), Color.Black);
619                     spriteBatch.DrawString(font, dexMod.ToString(), new
                    Vector2(500, 280), Color.Black);
620                     spriteBatch.DrawString(font, strMod.ToString(), new
                    Vector2(500, 335), Color.Black);
621
622                     if (Dex.text != "")
623                     {
624                         AC = 10 + (skillModifiers[Int32.Parse(Dex.text) /
                    2]);
625                     }
626                     spriteBatch.DrawString(font, "AC: " + AC, new Vector2
                    (650, 230), Color.Black);
627
628                     #endregion
629                     break;
630                 case GameState.LoadCharacter:
631                     #region LoadCharacter
632                     if (!validUser)
633                     {
634                         spriteBatch.DrawString(font, "Enter UserName:", new
                    Vector2(320, 110), Color.Black);
635                         Username.Update(mouseState, spriteBatch);
636                         enter.Update(mouseState, spriteBatch);
637                     }
638                     else
639                     {
640                         #region Load
641                         SQLiteConnection sqlite_conn;
642                         sqlite_conn = DBManager.CreateConnection();
643                         DBManager.ReadCharacters(sqlite_conn, spriteBatch,
                    font, Username.text);
644                         sqlite_conn.Close();
645                         Menu.Update(mouseState, spriteBatch);
646                         nextPage.Update(mouseState, spriteBatch);
647                         lastPage.Update(mouseState, spriteBatch);
648                         ID.Update(mouseState, spriteBatch);
649                         #endregion
650                     }
651                     #endregion
652                     break;
653                 case GameState.LoadWeapon:
654                     break;
655                 case GameState.Map: // if move to square with enemy it breaks
                     as the enemy cant move so it never becomes players turn
                    again
656                     #region movement on map
657                     #region draw map
658                     for (int y = 0; y < map.GetLength(0); y++)
```

```
659                     {
660                         for (int x = 0; x < map.GetLength(1); x++)
661                         {
662                             switch (map[y, x])
663                             {
664                                 case 0: //concrete
665                                     spriteBatch.Draw(concrete, new Rectangle ↪
                    (x * 40, y * 35, 40, 35), Color.White);
666                                     break;
667                                 case 1: //grass
668                                     spriteBatch.Draw(grass, new Rectangle(x * ↪
                     40, y * 35, 40, 35), Color.White);
669                                     break;
670                                 case 2: //sand
671                                     spriteBatch.Draw(sand, new Rectangle(x *  ↪
                    40, y * 35, 40, 35), Color.White);
672                                     break;
673                                 case 3: //water
674                                     spriteBatch.Draw(water, new Rectangle(x * ↪
                     40, y * 35, 40, 35), Color.White);
675                                     break;
676                                 case 4: //tree
677                                     spriteBatch.Draw(grass, new Rectangle(x * ↪
                     40, y * 35, 40, 35), Color.White);
678                                     spriteBatch.Draw(tree, new Rectangle(x *  ↪
                    40, y * 35, 40, 35), Color.White);
679                                     break;
680                                 case 5: //wall
681                                     spriteBatch.Draw(wall, new Rectangle(x *  ↪
                    40, y * 35, 40, 35), Color.White);
682                                     break;
683 
684                             }
685                         }
686                     }
687                     #endregion
688                     graph = BuildGraph(map);
689                     int[,] distances = graph.dijksta(PlayerLocation);
690 
691                     #region player/enemy location and movable squares
692                     for (int i = 0; i < distances.GetLength(0); i++)
693                     {
694                         int count = 0;
695                         int x = i;
696                         while (x > 19)
697                         {
698                             x -= 20;
699                             count++;
700                         }
701                         if (distances[i, 0] <= 30)
702                         {
703                             if (i != EnemyLocation && i != PlayerLocation)
704                             {
705                                 spriteBatch.Draw(greenSquare, new Rectangle(x ↪
                     * 40, count * 35, 40, 35), Color.White);
706                                 move[count, x] = 1;
```

```
707                             }
708                             else
709                             {
710                                 move[count, x] = 0;
711                                 playerX = x;
712                                 playerY = count;
713                                 spriteBatch.Draw(blueSquare, new Rectangle(x
                    * 40, count * 35, 40, 35), Color.White);
714                             }
715                         }
716                         else
717                         {
718                             move[count, x] = 0;
719                         }
720                     }
721 
722                     for (int y = 0; y < move.GetLength(0); y++)
723                     {
724                         for (int x = 0; x < move.GetLength(1); x++)
725                         {
726                             if (move[y, x] == 1)
727                             {
728                                 spriteBatch.Draw(greenSquare, new Rectangle(x
                     * 40, y * 35, 40, 35), Color.White);
729                             }
730                         }
731                     }
732                     spriteBatch.Draw(blueSquare, new Rectangle(playerX * 40,
                    playerY * 35, 40, 35), Color.White);
733                     #endregion
734                     UpdateEnemylocation();
735                     //breaks if you try to move to the square the enemy is on
736                     if (playersturn)
737                     {
738                         spriteBatch.Draw(redSquare, new Rectangle(5, 445, 10,
                     10), Color.White);
739                         #region player Move
740                         if (mouseState.LeftButton == ButtonState.Pressed &&
                    moved == false)
741                         {
742                             int x = mouseState.X / 40;
743                             int y = mouseState.Y / 35;
744                             try
745                             {
746                                 if (move[y, x] == 1)
747                                 {
748                                     playerX = x;
749                                     playerY = y;
750                                     PlayerLocation = y * map.GetLength(1) +
                    x;
751                                     moved = true;
752                                 }
753                             }
754                             catch { }
755                         }
756                         #endregion
```

```
757                         if (moved && actionTaken)
758                         {
759                             playersturn = false;
760                         }
761                     }
762                     else
763                     {
764                         spriteBatch.Draw(redSquare, new Rectangle(655, 445,
                    10, 10), Color.White);
765                         Thread.Sleep(1000);
766                         //enemy moves as far as possible along the shortest
                    path to the player
767                         #region enemy move
768                         int[,] distancesEnemy = graph.dijksta(EnemyLocation);
769                         int i = PlayerLocation;
770                         while (distancesEnemy[i, 0] > enemy.speed || i ==
                    PlayerLocation)
771                         {
772
773                             i = distancesEnemy[i, 1];
774                             EnemyLocation = i;
775                         }
776                         distancesEnemy = graph.dijksta(EnemyLocation);
777                         if (distancesEnemy[PlayerLocation,0] <= enemy.range
                    || distancesEnemy[PlayerLocation, 0] <= enemyweapon.range)
778                         {
779                             gameState = GameState.Fight;
780                         }
781                         else
782                         {
783                             playersturn = true;
784                             dodged = false;
785                         }
786
787                         #endregion
788                         moved = false;
789                         actionTaken = false;
790                     }
791                     #endregion
792                     #region text/buttons
793                     spriteBatch.DrawString(smallFont, "PlayerHP: " +
                    character.HP, new Vector2(20, 440), Color.Black);
794                     spriteBatch.DrawString(smallFont, "Action: " +
                    actionTaken, new Vector2(145, 430), Color.Black);
795                     spriteBatch.DrawString(smallFont, "Moved: " + moved, new
                    Vector2(145, 450), Color.Black);
796                     spriteBatch.DrawString(smallFont, "Actions:", new Vector2
                    (270, 440), Color.Black);
797                     attack.Update(mouseState, spriteBatch);
798                     dash.Update(mouseState, spriteBatch);
799                     dodge.Update(mouseState, spriteBatch);
800                     endTurn.Update(mouseState, spriteBatch);
801                     spriteBatch.DrawString(smallFont, "EnemyHP: " + enemy.HP,
                     new Vector2(670, 440), Color.Black);
802                     #endregion
803                     break;
```

```
804                 case GameState.Fight:
805                     //when attack chosen
806                     #region character info
807                     GraphicsDevice.Clear(Color.White);
808                     spriteBatch.Draw(awakenedShrub, new Rectangle(0, 0, 518 / 
                        3, 694 / 3), Color.White);
809                     spriteBatch.Draw(CharacterIcon, new Rectangle(563, 220, 
                        474 / 2, 505 / 2), Color.White);
810                     spriteBatch.DrawString(font, "PlayerHP: " + character.HP, 
                        new Vector2(540, 25), Color.Black);
811                     spriteBatch.DrawString(font, "AC: " + character.AC, new 
                        Vector2(540, 65), Color.Black);
812                     spriteBatch.DrawString(font, "Weapon: " + weapon.name, 
                        new Vector2(540, 105), Color.Black);
813                     spriteBatch.DrawString(font, "Range: " + weapon.range, 
                        new Vector2(540, 145), Color.Black);
814                     if (weapon.throwable)
815                     {
816                         spriteBatch.DrawString(font, "(thrown)", new Vector2 
                        (690, 145), Color.Black);
817                     }
818                     spriteBatch.DrawString(font, "EnemyHP: " + enemy.HP, new 
                        Vector2(10, 275), Color.Black);
819                     spriteBatch.DrawString(font, "EnemyAC: " + enemy.AC, new 
                        Vector2(10, 315), Color.Black);
820                     if (enemyweapon.name != "")
821                     {
822                         spriteBatch.DrawString(font, "Weapon: " + 
                        enemyweapon.name, new Vector2(10, 355), Color.Black);
823                         spriteBatch.DrawString(font, "Range: " + 
                        enemyweapon.range, new Vector2(10, 395), Color.Black);
824                         if (enemyweapon.throwable)
825                         {
826                             spriteBatch.DrawString(font, "(thrown)", new 
                        Vector2(160, 395), Color.Black);
827                         }
828                     }
829                     #endregion
830                     #region players turn
831                     if (playersturn)
832                     {
833                         int startHP = enemy.HP;
834                         distances = graph.dijksta(PlayerLocation);
835                         spriteBatch.DrawString(font, "Distance: " + distances
                        [EnemyLocation, 0], new Vector2(200, 40), Color.Black);
836                         bool thrown = false;
837                         if (weapon.inRange(EnemyLocation, PlayerLocation, 
                        graph, ref thrown))
838                         {
839                             spriteBatch.DrawString(font, "Enemy is in 
                        range.", new Vector2(200, 80), Color.Black);
840                             spriteBatch.DrawString(font, "Roll attack die:", 
                        new Vector2(200, 140), Color.Black);
841                             d20.Update(mouseState, spriteBatch);
842                             int attackroll = 0;
843                             Int32.TryParse(d20.text, out attackroll);
```

```
844                         if (attackroll != 0 && attackroll >= enemy.AC)
845                         {
846                             spriteBatch.DrawString(font, "attack die is greater than AC", new Vector2(200, 200), Color.Black);
847                             spriteBatch.DrawString(font, "Enemy is hit, roll hitDie (d" + weapon.hitDie + ")", new Vector2(200, 240), Color.Black);
848                             hitDie.Update(mouseState, spriteBatch);
849                             int damage = 0;
850                             Int32.TryParse(hitDie.text, out damage);
851                             if (damage != 0)
852                             {
853                                 spriteBatch.DrawString(font, "Enemy has taken " + damage + " damage", new Vector2(200, 360), Color.Black);
854                                 if (!damageDone)
855                                 {
856                                     enemy.HP = startHP - damage;
857                                     damageDone = true;
858                                     if (enemy.HP <= 0)
859                                     {
860                                         gameState = GameState.GameOver;
861                                     }
862                                 }
863                                 ReturnToMap.Update(mouseState, spriteBatch);
864                             }
865                         }
866                         else if (attackroll != 0 && attackroll < enemy.AC)
867                         {
868                             spriteBatch.DrawString(font, "attack die is less than AC", new Vector2(200, 200), Color.Black);
869                             spriteBatch.DrawString(font, "Enemy takes no damage", new Vector2(200, 240), Color.Black);
870                             ReturnToMap.Update(mouseState, spriteBatch);
871                         }
872 
873                     }
874                     else
875                     {
876                         spriteBatch.DrawString(font, "Enemy not in range.", new Vector2(200, 80), Color.Black);
877                         actionTaken = false;
878                         ReturnToMap.Update(mouseState, spriteBatch);
879                     }
880                 }
881                 #endregion
882                 #region enemys turn
883                 else
884                 {
885                     int maxDamage;
886                     if(enemy.hasWeapon)
887                     {
888                         maxDamage = enemyweapon.hitDie;
889                     }
```

```
890                         else
891                         {
892                             maxDamage = (int)enemy.hitDie;
893                         }
894                         int startHP = character.HP;
895                         spriteBatch.DrawString(font, "Attack die:", new Vector2(200, 40), Color.Black);
896 
897                         if (dodged)
898                         {
899                             if(!attackdieRolled)
900                             {
901                                 roll1 = rand.Next(1, 21);
902                                 roll2 = rand.Next(1, 21);
903                                 attackdieRolled = true;
904                             }
905                             spriteBatch.DrawString(smallFont, "You dodged, enemy has disadvantage", new Vector2(200, 80), Color.DarkRed);
906                             spriteBatch.DrawString(smallFont, "Roll 1:", new Vector2(200, 150), Color.DarkRed);
907                             spriteBatch.Draw(d20Image, new Rectangle(265, 120, 80, 80), Color.White);
908                             spriteBatch.DrawString(smallFont, "Roll 2:", new Vector2(350, 150), Color.DarkRed);
909                             spriteBatch.Draw(d20Image, new Rectangle(415, 120, 80, 80), Color.White);
910                             spriteBatch.DrawString(smallFont, roll1.ToString(), new Vector2(295, 150), Color.Black);
911                             spriteBatch.DrawString(smallFont, roll2.ToString(), new Vector2(445, 150), Color.Black);
912                             if(roll1 < roll2)
913                             {
914                                 attackroll = roll1;
915                             }
916                             else
917                             {
918                                 attackroll = roll2;
919                             }
920                         }
921                         else
922                         {
923                             spriteBatch.Draw(d20Image, new Rectangle(265, 100, 80, 80), Color.White);
924                             if (!attackdieRolled)
925                             {
926                                 attackroll = rand.Next(1, 21);
927                                 attackdieRolled = true;
928                             }
929                             spriteBatch.DrawString(smallFont, attackroll.ToString(), new Vector2(295, 130), Color.Black);
930                         }
931                         if (attackroll >= character.AC)
932                         {
933                             spriteBatch.DrawString(font, "attack die is
```

```
                    greater than AC", new Vector2(200, 200), Color.Black);
934                         spriteBatch.DrawString(font, "You are hit, enemy
                    hitDie: (d" + maxDamage + ")", new Vector2(200, 240),
                    Color.Black);
935                         if(!HitDieRolled)
936                         {
937                             playerDamage = rand.Next(1, maxDamage + 1);
938                             HitDieRolled = true;
939                         }
940                         spriteBatch.DrawString(smallFont, "HitDie:", new
                    Vector2(200, 320), Color.DarkRed);
941                         spriteBatch.Draw(d20Image, new Rectangle(275,
                    285, 80, 80), Color.White);
942                         spriteBatch.DrawString(smallFont,
                    playerDamage.ToString(), new Vector2(305, 310),
                    Color.Black);
943
944                         if (playerDamage != 0)
945                         {
946                             spriteBatch.DrawString(font, "You have taken
                    " + playerDamage + " damage", new Vector2(200, 370),
                    Color.Black);
947                             if (!damageDone)
948                             {
949                                 character.HP = startHP - playerDamage;
950                                 damageDone = true;
951                                 if (character.HP <= 0)
952                                 {
953                                     gameState = GameState.GameOver;
954                                 }
955                             }
956                             ReturnToMap.Update(mouseState, spriteBatch);
957                         }
958                     }
959                     else if (attackroll != 0 && attackroll <
                    character.AC)
960                     {
961                         spriteBatch.DrawString(font, "attack die is less
                    than AC", new Vector2(200, 200), Color.Black);
962                         spriteBatch.DrawString(font, "Enemy does no
                    damage", new Vector2(200, 240), Color.Black);
963                         ReturnToMap.Update(mouseState, spriteBatch);
964                     }
965                 }
966                 #endregion
967                 break;
968             case GameState.GameOver:
969                 spriteBatch.DrawString(font, "GAME OVER", new Vector2
                    (200, 80), Color.Black);
970                 if(enemy.HP <= 0)
971                 {
972                     spriteBatch.DrawString(font, "You Win!", new Vector2
                    (200, 120), Color.Black);
973                 }
974                 else
975                 {
```

```
976                         spriteBatch.DrawString(font, "You Lose :(", new
                        Vector2(200, 120), Color.Black);
977                     }
978                     Menu.Update(mouseState, spriteBatch);
979                     break;
980             }
981
982             spriteBatch.End();
983             base.Draw(gameTime);
984         }
985
986         public void UpdateEnemylocation()
987         {
988             int x = EnemyLocation;
989             int y = 0;
990             while (x > 19)
991             {
992                 x -= 20;
993                 y++;
994             }
995             spriteBatch.Draw(redSquare, new Rectangle(x * 40, y * 35, 40,
                35), Color.White);
996         }
997
998
999         Graph BuildGraph(int[,] map)
1000        {
1001            // multipliers for different terrain types (concrete, grass,
                sand, water, tree, wall)
1002            double[] multiplier = new double[] { 1, 1.2, 1.8, 2.2, 0, 0 };
1003            List<int> unvisitable = new List<int>();
1004            #region addNodes
1005            Graph graph = new Graph(map.Length);
1006            for (int a = 0; a < map.GetLength(0); a++)
1007            {
1008                for (int b = 0; b < map.GetLength(1); b++)
1009                {
1010                    string nodeName = a.ToString() + "," + b.ToString();
1011                    graph.AddNode(nodeName);
1012                }
1013
1014            }
1015            #endregion
1016
1017            #region addEdges
1018            for (int y = 0; y < map.GetLength(0); y++)
1019            {
1020                for (int x = 0; x < map.GetLength(1); x++)
1021                {
1022                    if (multiplier[map[y, x]] == 0)
1023                    {
1024                        unvisitable.Add((y * map.GetLength(1)) + x);
1025                    }
1026                    else
1027                    {
1028
```

```
1029                        if (y > 0) //y-1,
1030                        {
1031                            graph.AddEdge((y * map.GetLength(1)) + x, ((y -
                        1) * map.GetLength(1)) + x, (int)(5.0 * multiplier[map[y,
                        x]]), 1);
1032                        }
1033                        if (y < map.GetLength(0) - 1) //y+1
1034                        {
1035                            graph.AddEdge((y * map.GetLength(1)) + x, ((y +
                        1) * map.GetLength(1)) + x, (int)(5.0 * multiplier[map[y,
                        x]]), 1);
1036                        }
1037                        if (x > 0) //x-1
1038                        {
1039                            graph.AddEdge((y * map.GetLength(1)) + x, (y *
                        map.GetLength(1)) + x - 1, (int)(5.0 * multiplier[map[y,
                        x]]), 1);
1040                        }
1041                        if (x < map.GetLength(1) - 1) //x+1
1042                        {
1043                            graph.AddEdge((y * map.GetLength(1)) + x, (y *
                        map.GetLength(1)) + x + 1, (int)(5.0 * multiplier[map[y,
                        x]]), 1);
1044                        }
1045                    }
1046                }
1047            }
1048            //remove edges of where nodes are unvisitable (tree/wall)
1049            foreach (int n in unvisitable)
1050            {
1051                for (int i = 0; i < map.Length; i++)
1052                {
1053                    graph.RemoveEdge(n, i);
1054                }
1055            }
1056            #endregion
1057            return graph;
1058        }
1059    }
1060 }
1061
```