```
 1  using BasicAI;
 2  using Final_Project.Properties;
 3  using Microsoft.VisualBasic.Devices;
 4  using System.Transactions;
 5
 6  //programmer: Aaron Yoon
 7  //project: Final Project
 8  //Date: 5/25/23
 9  namespace Final_Project
10  {
11      public partial class Form1 : Form
12      {
13
14          private static Form1 instance = new Form1();
15
16          //declare all variables
17          int playerSpeed = 10;
18          int test = 0;
19          String movementKey = "";
20          String modifierKey = "";
21
22          int kills = 0;
23
24          List<PictureBox> heartList = new List<PictureBox>();
25          int heartCount = 0;
26
27          Label[] projectiles = new Label[5];
28          int bulletCount = 0;
29
30          Boolean timeFrozen = false;
31          Boolean timeRegen = false;
32
33          System.Random r = new System.Random((int)
              System.DateTime.Now.Ticks);
34
35          //variable to control enemies
36          Enemy[] enemies = new Enemy[5];
37
38          Enemy enemy0;
39          Enemy enemy1;
40          Enemy enemy2;
41          Enemy enemy3;
42          Enemy enemy4;
43
44          Gunner boss;
45          RoundState currentRound = RoundState.ROUND_1;
46          WeaponSelected currentWeapon = WeaponSelected.SWORD;
47
48          //a way to show what the current round is
```

```
49          private enum RoundState
50          {
51              ROUND_1,
52              ROUND_2,
53              ROUND_3,
54              BOSS
55          }
56
57          //a way to show the current weapon selected
58          public enum WeaponSelected
59          {
60              SWORD,
61              SHIELD,
62          }
63          public Form1()
64          {
65
66              InitializeComponent();
67
68              //enemy0 = new Enemy(picEnemy0, player, projectiles);
69
70              //private static Form1 instance = new Form1();
71
72              //give values to all variables
73              projectiles[0] = projectile1;
74              projectiles[1] = projectile2;
75              projectiles[2] = projectile3;
76              projectiles[3] = projectile4;
77              projectiles[4] = projectile5;
78
79              enemy0 = new Enemy(picEnemy0, player, projectiles, 0.015, 3);
80              enemy1 = new Enemy(picEnemy1, player, projectiles, 0.015, 3);
81              enemy2 = new Enemy(picEnemy2, player, projectiles, 0.015, 3);
82              enemy3 = new Enemy(picEnemy3, player, projectiles, 0.015, 3);
83              enemy4 = new Enemy(picEnemy4, player, projectiles, 0.015, 3);
84
85              boss = new Gunner(picBoss, player, bossBullet, projectiles,
                  bossHealthBar, 0.013, 400, health);
86
87              boss.width = this.Width;
88              boss.height = this.Height;
89
90              enemies[0] = enemy0;
91              enemies[1] = enemy1;
92              enemies[2] = enemy2;
93              enemies[3] = enemy3;
94              enemies[4] = enemy4;
95
96
```

```
 97              //adjustEnemies(1, 0.015);
 98              initRound();
 99          }
100
101          public static Form1 getInstance()
102          {
103              return instance;
104          }
105          private void Form1_Load(object sender, EventArgs e)
106          {
107              //boss.respawn(0, 0);
108
109              Update.Enabled = true;
110              itemTimer.Enabled = true;
111
112              stamina.Width = 400;
113              health.Width = 400;
114
115
116          }
117
118          private void movePlayer(String key)
119          {
120              //makes the player move depending on which key and if
                   sprinting
121
122              if (modifierKey == "Shift" && stamina.Width != 0)
123              {
124                  staminaTimer.Enabled = false;
125                  playerSpeed = 20;
126                  stamina.Width -= 4;
127              }
128              else if (modifierKey == "None" && stamina.Width != 0)
129              {
130                  staminaTimer.Enabled = true;
131                  playerSpeed = 10;
132              }
133              else if (stamina.Width == 0)
134              {
135                  staminaTimer.Enabled = true;
136                  playerSpeed = 5;
137              }
138
139
140              switch (key)
141              {
142                  case "w":
143                      player.Top -= playerSpeed;
144                      break;
```

```
145                 case "a":
146                     player.Left -= playerSpeed;
147                     break;
148                 case "s":
149                     player.Top += playerSpeed;
150                     break;
151                 case "d":
152                     player.Left += playerSpeed;
153                     break;
154 
155                 case "W":
156                     player.Top -= playerSpeed;
157                     break;
158                 case "A":
159                     player.Left -= playerSpeed;
160                     break;
161                 case "S":
162                     player.Top += playerSpeed;
163                     break;
164                 case "D":
165                     player.Left += playerSpeed;
166                     break;
167 
168 
169             }
170         }
171 
172         private double getAngle()
173         {
174             //used to get the angle of your cursor in relation to the      ↵
                  player
175             int mouseX = Cursor.Position.X;
176             int mouseY = Cursor.Position.Y;
177 
178             int playerX = player.Left + (player.Width / 2);
179             int playerY = player.Top + (player.Height / 2) + 30;
180 
181             double deltaX = mouseX - playerX;
182             double deltaY = playerY - mouseY;
183 
184             try
185             {
186 
187             double angle = Math.Atan(deltaY / deltaX) + (Math.PI / 2);
188 
189                 if (deltaX < 0)
190                     return Math.PI + angle;
191                 else
192                     return angle;
```

```
193
194             }
195             catch
196             {
197                 return 0;
198             }
199
200         }
201
202         private void drawArc(double angle)
203         {
204             //draws an arc (shield) around the player
205             double referenceAngle = angle - (Math.PI / 10);
206
207             double x;
208             double y;
209
210             try
211             {
212                 for (int i = 0; i < projectiles.Length; i++)
213                 {
214                     x = Math.Sin(referenceAngle);
215                     y = Math.Cos(referenceAngle);
216
217                     projectiles[i].Left = player.Left + (player.Width / 2) ↪
                    + (int)(x * 100);
218                     projectiles[i].Top = player.Top + (player.Height / 2)  ↪
                   + (int)((y * 100));
219
220                     referenceAngle += Math.PI / 20;
221                 }
222             }
223             catch { }
224
225             //label1.Text = x.ToString() + "\n" + y.ToString();
226         }
227
228         private void drawSword(double angle)
229         {
230             //draws a stick (sword) around the player
231             double x = Math.Sin(angle);
232             double y = Math.Cos(angle);
233
234             try
235             {
236                 for (int i = 0; i < projectiles.Length; i++)
237                 {
238
239                     projectiles[i].Left = player.Left + (player.Width / 2) ↪
```

```
                        + (int)(x * 100);
240                     projectiles[i].Top = player.Top + (player.Height / 2)
                       + (int)(y * 100);
241
242                     //label1.Text = x.ToString() + "\n" + y.ToString();
243                     x *= 1.1;
244                     y *= 1.1;
245                 }
246             }
247             catch { }
248         }
249
250         //
              ================================================================
              ============================================================
251         private void timer1_Tick(object sender, EventArgs e)
252         {
253             //update timer for everything
254             try
255             {
256                 movePlayer(movementKey);
257                 //drawArc(getAngle());
258
259                 if(currentWeapon == WeaponSelected.SWORD)
260                     drawSword(getAngle());
261                 else if(currentWeapon == WeaponSelected.SHIELD)
262                     drawArc(getAngle());
263
264                 heal();
265
266
267                 label1.Text = getKills().ToString();
268
269                 if(!timeFrozen)
270                     updateEnemies();
271
272                 if (currentRound == RoundState.BOSS)
273                 {
274
275                     boss.Update();
276                     boss.dynamicWaypoint(this.Width, this.Height);
277
278                     if (boss.getHealth() == 0)
279                     {
280
281                         btnReset.Visible = true;
282                         btnReset.Enabled = true;
283                         Update.Enabled = false;
284
```

```
285                         MessageBox.Show("The tyrant has been slain!");
286
287                     }
288                 }
289
290                 changeRound();
291             }
292
293             catch
294             {
295                 MessageBox.Show("How did we get here?");
296             }
297         }
298         //=============================================================
            ============================================================
299
300
301         private void changeRound()
302         {
303             //change round based off of how many kills you have
304             if(currentRound == RoundState.ROUND_1 && getKills() >= 10)
305             {
306                 currentRound = RoundState.ROUND_2;
307                 initRound();
308             }
309             else if(currentRound == RoundState.ROUND_2 && getKills() >=
                  25)
310             {
311                 currentRound = RoundState.ROUND_3;
312                 initRound();
313             }
314             else if (currentRound == RoundState.ROUND_3 && getKills() >=
                  45)
315             {
316                 currentRound = RoundState.BOSS;
317                 initRound();
318             }
319
320
321         }
322
323         private void initRound()
324         {
325             //initialize the round for what it needs
326             switch(currentRound)
327             {
328                 case RoundState.ROUND_1:
329                     adjustEnemies(2, 0.015);
```

```
330                     RoundIndicator.BackColor = Color.Lime;
331                     changeImage("happy.png");
332                     break;
333                 case RoundState.ROUND_2:
334                     adjustEnemies(3, 0.020);
335                     RoundIndicator.BackColor = Color.Orange;
336                     changeImage("meh.png");
337                     break;
338                 case RoundState.ROUND_3:
339                     adjustEnemies(4, 0.03);
340                     RoundIndicator.BackColor = Color.Red;
341                     changeImage("angy.png");
342                     break;
343                 case RoundState.BOSS:
344                     adjustEnemies(4, 0.03);
345                     RoundIndicator.BackColor = Color.Black;
346                     changeImage("demon.png");
347                     arbi.Visible = true;
348                     boss.respawn(0, 0);
349                     break;
350             }
351         }
352 
353         private void changeImage(String name)
354         {
355             //used to mass change the images of all enemies
356             for(int i = 0; i < enemies.Length; i++)
357             {
358                 enemies[i].picEnemy.Image = Image.FromFile(name, true);
359             }
360         }
361 
362         private void adjustEnemies(int cap, double spd)
363         {
364             //change the speed and health of the enemies
365             for(int i = 0; i < enemies.Length; i++)
366             {
367                 enemies[i].healthcap = cap;
368                 enemies[i].speed = spd;
369             }
370         }
371 
372         private void refillStamina()
373         {
374             //used to refill the stamina
375             //eventTimer.Enabled = true;
376             staminaTimer.Interval = 40;
377 
378             if (stamina.Width < 400)
```

```
379                 stamina.Width += 10;
380             else
381             {
382                 staminaTimer.Enabled = false;
383                 staminaTimer.Interval = 1500;
384             }
385 
386         }
387 
388         private void heal()
389         {
390             //used to automatically check if you're able to grab a heart
                  to heal
391             for(int i = 0; i < heartList.Count; i++)
392             {
393                 if (player.Bounds.IntersectsWith(heartList.ElementAt
                      (i).Bounds) && health.Width < 400)
394                 {
395                     health.Width += 40;
396                     this.Controls.Remove(heartList.ElementAt(i));
397                     heartList.RemoveAt(i);
398                     heartCount--;
399                 }
400             }
401         }
402 
403         private void updateEnemies()
404         {
405             //a way to mass update all enemies and to automatically take
                  damage
406             for (int i = 0; i < enemies.Length; i++)
407             {
408                 if (enemies[i].enemyTouch())
409                 {
410                     health.Width -= 80;
411                 }
412 
413                 enemies[i].Update();
414             }
415 
416             if (health.Width <= 0)
417             {
418                 btnReset.Visible = true;
419                 btnReset.Enabled = true;
420                 Update.Enabled = false;
421 
422                 MessageBox.Show("Game Over!");
423             }
424         }
```

```
425
426         private double getKills()
427         {
428             //displays the amount of kills you have
429             kills = 0;
430
431             for (int i = 0; i < enemies.Length; i++)
432             {
433                 kills += enemies[i].getDeaths();
434             }
435
436             return kills;
437             //label1.Text = kills.ToString();//enemy0.getHealth().ToString
                  ();//enemy0.swordTouch().ToString();
438         }
439         public void spawnHeart()
440         {
441             //gives a random chance to spawn a heart
442             heartList.Add(new PictureBox());
443
444             this.Controls.Add(heartList.ElementAt(heartCount));
445             heartList.ElementAt(heartCount).Height = 50;
446             heartList.ElementAt(heartCount).Width = 50;
447             heartList.ElementAt(heartCount).SizeMode =
                  PictureBoxSizeMode.StretchImage;
448             heartList.ElementAt(heartCount).Image = Image.FromFile
                  ("heart.png", true);
449             heartList.ElementAt(heartCount).Left = r.Next(0, this.Width -
                  50);
450             heartList.ElementAt(heartCount).Top = r.Next(0, this.Height -
                  50);
451
452             heartCount++;
453         }
454         private void Form1_KeyDown(object sender, KeyEventArgs e)
455         {
456             modifierKey = Control.ModifierKeys.ToString();
457
458
459         }
460
461         private void reset()
462         {
463             //reset everything back to round 1
464             currentRound = RoundState.ROUND_1;
465             initRound();
466
467             boss.reset();
468             arbi.Visible = false;
```

```
469
470             health.Width = 400;
471
472             for(int i = 0; i < enemies.Length; i++)
473             {
474                 enemies[i].reset();
475             }
476
477             btnReset.Visible = false;
478             btnReset.Enabled = false;
479
480             Update.Enabled = true;
481
482
483         }
484
485         private void player_Click(object sender, EventArgs e)
486         {
487
488         }
489
490         private void Form1_KeyUp(object sender, KeyEventArgs e)
491         {
492             //movementKey = 0;
493             modifierKey = Control.ModifierKeys.ToString();
494
495             String keyValue = "";
496
497             if (modifierKey == "Shift")
498             {
499
500                 switch (e.KeyValue)
501                 {
502                     case 87:
503                         keyValue = "W";
504                         break;
505                     case 65:
506                         keyValue = "A";
507                         break;
508                     case 83:
509                         keyValue = "S";
510                         break;
511                     case 68:
512                         keyValue = "D";
513                         break;
514                 }
515             }
516
517             else if (modifierKey == "None")
```

```
518             {
519 
520                 switch (e.KeyValue)
521                 {
522                     case 87:
523                         keyValue = "w";
524                         break;
525                     case 65:
526                         keyValue = "a";
527                         break;
528                     case 83:
529                         keyValue = "s";
530                         break;
531                     case 68:
532                         keyValue = "d";
533                         break;
534                 }
535             }
536 
537             if (movementKey == keyValue)
538             {
539                 movementKey = "";
540             }
541         }
542 
543         private void Form1_KeyPress(object sender, KeyPressEventArgs e)
544         {
545 
546 
547             movementKey = e.KeyChar.ToString();
548 
549             //label1.Text = movementKey;
550             //label1.Text = Control.ModifierKeys.ToString();
551         }
552 
553         private void Form1_PreviewKeyDown(object sender,                ↵
              PreviewKeyDownEventArgs e)
554         {
555 
556         }
557 
558         private void stamina_Click(object sender, EventArgs e)
559         {
560 
561         }
562 
563         private void staminaTimer_Tick(object sender, EventArgs e)
564         {
565             refillStamina();
```

```
566         }
567
568         private void Form1_MouseClick(object sender, MouseEventArgs e)
569         {
570             //change weapons if you click
571             if (currentWeapon == WeaponSelected.SWORD)
572             {
573                 currentWeapon = WeaponSelected.SHIELD;
574                 picWeapon.Image = (Image)Resources.shield;
575
576                 for(int i = 0; i < enemies.Length; i++)
577                 {
578                     enemies[i].currentWeapon = currentWeapon;
579
580                 }
581
582                 boss.currentWeapon = currentWeapon;
583             }
584             else if (currentWeapon == WeaponSelected.SHIELD)
585             {
586                 currentWeapon = WeaponSelected.SWORD;
587                 picWeapon.Image = (Image)Resources.sword1;
588                 for (int i = 0; i < enemies.Length; i++)
589                 {
590                     enemies[i].currentWeapon = currentWeapon;
591
592                 }
593
594                 boss.currentWeapon = currentWeapon;
595             }
596         }
597
598         private void itemTimer_Tick(object sender, EventArgs e)
599         {
600             //used to randomly spawn a heart
601             int proc = r.Next(0, 100);
602
603             if(proc == 3)
604             {
605                 spawnHeart();
606             }
607         }
608
609         private void quitToolStripMenuItem_Click(object sender, EventArgs
              e)
610         {
611             this.Close();
612         }
613
```

```
614         private void fileToolStripMenuItem_Click(object sender, EventArgs
              e)
615         {
616             //Update.Enabled = false;
617         }
618
619         private void pauseUnpauseToolStripMenuItem_Click(object sender,
              EventArgs e)
620         {
621             //toggle update timer
622             if (Update.Enabled)
623                 Update.Enabled = false;
624             else
625                 Update.Enabled = true;
626         }
627
628         private void button1_Click(object sender, EventArgs e)
629         {
630
631         }
632
633         private void yBox_TextChanged(object sender, EventArgs e)
634         {
635
636         }
637
638         private void xBox_TextChanged(object sender, EventArgs e)
639         {
640
641         }
642
643         private void Form1_MouseDoubleClick(object sender, MouseEventArgs
              e)
644         {
645             if (timeIcon.Height >= 150)
646             {
647
648                 stopTime.Enabled = true;
649             }
650         }
651
652         private void stopTime_Tick(object sender, EventArgs e)
653         {
654             //used to stop time
655             if (!timeRegen)
656             {
657                 timeFrozen = true;
658
659                 timeIcon.Height -= 5;
```

```
660             }
661
662             if (timeRegen && timeIcon.Height < 150)
663             {
664                 timeIcon.Height += 2;
665             }
666             else if(timeIcon.Height >= 150)
667             {
668                 timeRegen = false;
669                 stopTime.Enabled = false;
670             }
671
672             if(timeIcon.Height <= 0)
673             {
674                 timeFrozen = false;
675                 timeRegen = true;
676             }
677         }
678
679         private void bossHealthBar_Click(object sender, EventArgs e)
680         {
681
682         }
683
684         private void btnReset_Click(object sender, EventArgs e)
685         {
686             reset();
687         }
688
689         private void btnReset_Click_1(object sender, EventArgs e)
690         {
691             reset();
692         }
693
694         private void newGameToolStripMenuItem_Click(object sender,
              EventArgs e)
695         {
696             reset();
697         }
698
699         private void aboutToolStripMenuItem_Click(object sender, EventArgs
              e)
700         {
701             //when about is clicked, this message box shows
702             MessageBox.Show(
703                 "Hello! This game simulates you, a knight armed with a
                      sword and shield, defending his homeland against the
                      evil monsters!" + "\n"
704                 + "Use WASD to move and shift to sprint, though becareful!
```

```
                    Once you're out of stamina you become exhausted and
                    cannot walk as fast." + "\n"
705                 + "Use your mouse to control where you sword / shield is,
                    and left click to swap weapons. Double click to stop
                    time briefly." + "\n"
706                 + "overtime, there are chances for hearts to spawn around
                    the map. Pick them up for a small health boost." + "\n"
707                 + "That's all you need to know, now beat that tyrant!"
708                 );
709         }
710     }
711 }
```