```
 1
 2  using SplashKitSDK;
 3  using System.Security.Cryptography.X509Certificates;
 4
 5  namespace NitsMercernary
 6  {
 7      public class Program
 8      {
 9          private enum RoomType
10          {
11              JohnStreet,
12              Library,
13              BA,
14              ATC,
15              Dungeon,
16              Winner
17          }
18
19          private enum PlayerType
20          {
21              cartoon,
22              witch
23          }
24          public static void Main()
25          {
26              Player player;
27              Window window = new Window("NitsAdeventure", 600, 600);
28
29              int score = 0;
30
31              bool Entered;
32
33              Font font = SplashKit.LoadFont("score", "fonts/arial.ttf");
34              Music BackgroundMusic = SplashKit.LoadMusic("background",
                  "songs/song.mp3");
35              SplashKit.PlayMusic(BackgroundMusic, -1);
36
37              RoomType CurrentRoomType = RoomType.JohnStreet;
38              PlayerType CurrentPlayerType = PlayerType.cartoon;
39
40              WitchPlayerAnimation witchplayerani = new("witch", "witch
                  player animation", 300, 550);
41              CartoonPlayerAnimation cartoonplayerani = new("cartoon",
                  "cartoon player animation", 300, 550);
42              EnemyAnimation enemyani = new("enemy", "enemy animation", 550,
                   540); //enemy in ATC
43              EnemyAnimation enemyani2 = new("enemy", "enemy animation", 10,
                   340); //enemy in ATC
44              EnemyAnimation enemyani3 = new("enemy", "enemy animation",
```

```
                550, 373); //enemy in library
45              EnemyAnimation enemyani4 = new("enemy", "enemy animation",
                300, 430); // enemy in BA
46              EnemyAnimation enemyani5 = new("enemy", "enemy animation",
                300, 490); // enemy in Dungeon
47              List<EnemyAnimation> enemies = new List<EnemyAnimation>();
48              enemies.Add(enemyani);
49              enemies.Add(enemyani2);
50              enemies.Add(enemyani3);
51              enemies.Add(enemyani4);
52              enemies.Add(enemyani5);
53
54              Room JohnStreet = new JohnStreet(new string[]
                { "JohnStreet" }, "JohnStreet");
55              Room Library = new Library(new string[] { "Library" },
                "Library");
56              Room ATC = new ATC(new string[] { "ATC" }, "ATC");
57              Room BA = new BA(new string[] { "BA" }, "BA");
58              Room Dungeon = new Dungeon(new string[] { "Dungeon" },
                "Dungeon");
59              Room Winner = new Winner(new string[] { "Winner" }, "Winner");
60
61              // to create a point between player and enemy to discern
                distance between them
62              Player player1 = new("witchboundary","witchboundary",0,0);
63              player = cartoonplayerani;
64
65              PlayerInterface playerInterface;
66              playerInterface = player1;
67              do
68              {
69                  Room CurrentRoom;
70                  player1.X = witchplayerani.X + 20;
71
72                  //Keep track if a user has entered or exited a room
73                  Entered = false;
74
75                  SplashKit.ProcessEvents();
76                  SplashKit.ClearScreen();
77                  window.Clear(Color.White);
78                  CurrentRoom = new JohnStreet(new string[]
                    { "JohnStreet" }, "JohnStreet");
79
80                  //diplay score
81                  string ScoreText = $"GOLD {score.ToString()}G";
82
83
84                  //if the player enters a certain coordiantes it changes
                      current room to the new room it entered or exited
```

```
85                  if ((cartoonplayerani.PlayerIsAt(270, 90, 380, 0)) &&
                      (CurrentRoomType == RoomType.JohnStreet))
86                  {
87                      Entered = true;
88                      CurrentRoomType = RoomType.Library;
89                  }
90                  if ((cartoonplayerani.PlayerIsAt(0, 320, 110, 120)) &&
                      (CurrentRoomType == RoomType.JohnStreet))
91                  {
92                      Entered = true;
93                      CurrentRoomType = RoomType.BA;
94                  }
95                  if ((cartoonplayerani.PlayerIsAt(490, 320, 600, 120)) &&
                      (CurrentRoomType == RoomType.JohnStreet))
96                  {
97                      Entered = true;
98                      CurrentRoomType = RoomType.ATC;
99                  }
100                 if ((cartoonplayerani.PlayerIsAt(380, 394, 470, 340)) &&
                      (CurrentRoomType == RoomType.JohnStreet))
101                 {
102                     Entered = true;
103                     CurrentRoomType = RoomType.Dungeon;
104                 }
105                 // exits room to go back to john street
106                 if ((witchplayerani.PlayerIsAt(97, 530, 127, 446)) &&
                      (CurrentRoomType == RoomType.ATC) && (SplashKit.KeyDown
                      (KeyCode.SpaceKey)))
107                 {
108                     Entered = true;
109                     CurrentRoomType = RoomType.JohnStreet;
110                     CurrentRoom.PlayerStartingPos(cartoonplayerani, 470,
                       300);
111                 }
112                 if ((witchplayerani.PlayerIsAt(20, 420, 60, 370)) &&
                      (CurrentRoomType == RoomType.BA) && (SplashKit.KeyDown
                      (KeyCode.SpaceKey)))
113                 {
114                     Entered = true;
115                     CurrentRoomType = RoomType.JohnStreet;
116                     CurrentRoom.PlayerStartingPos(cartoonplayerani, 131,
                       228);
117                 }
118                 if ((witchplayerani.PlayerIsAt(9, 363, 39, 310)) &&
                      (CurrentRoomType == RoomType.Library) &&
                      (SplashKit.KeyDown(KeyCode.SpaceKey)))
119                 {
120                     Entered = true;
121                     CurrentRoomType = RoomType.JohnStreet;
```

```
122                     CurrentRoom.PlayerStartingPos(cartoonplayerani, 335,
                    99);
123                 }
124                 if ((witchplayerani.PlayerIsAt(50, 480, 90, 430)) &&
                    (CurrentRoomType == RoomType.Dungeon) &&
                    (SplashKit.KeyDown(KeyCode.SpaceKey)))
125                 {
126                     Entered = true;
127                     CurrentRoomType = RoomType.JohnStreet;
128                     CurrentRoom.PlayerStartingPos(cartoonplayerani, 378,
                    412);
129                 }
130                 if (score >= 100)
131                 {
132                     Entered = true;
133                     CurrentRoomType = RoomType.Winner;
134                 }
135 
136                 //when a new room is assigned it changes the room object
                    and player type object
137                 switch (CurrentRoomType)
138                 {
139                     case RoomType.JohnStreet:
140                         CurrentRoom = JohnStreet;
141                         CurrentPlayerType = PlayerType.cartoon;
142                         player.Room = CurrentRoom;
143                         break;
144                     case RoomType.Library:
145                         CurrentRoom = Library;
146                         CurrentPlayerType = PlayerType.witch;
147                         player.Room = enemyani3.Room = CurrentRoom;
148                         if (Entered) { CurrentRoom.PlayerStartingPos
                    (witchplayerani, 29, 363); }
149                         break;
150                     case RoomType.ATC:
151                         CurrentRoom = ATC;
152                         CurrentPlayerType = PlayerType.witch;
153                         player.Room = enemyani.Room = enemyani2.Room
                    =CurrentRoom;
154                         if (Entered) { CurrentRoom.PlayerStartingPos
                    (witchplayerani, 100, 530);}
155                         break;
156                     case RoomType.BA:
157 
158                         CurrentRoom = BA;
159                         CurrentPlayerType = PlayerType.witch;
160                         player.Room = enemyani4.Room = CurrentRoom;
161                         if (Entered) { CurrentRoom.PlayerStartingPos
                    (witchplayerani, 20, 420);}
```

```
162                         break;
163                     case RoomType.Dungeon:
164                         CurrentRoom = Dungeon;
165                         CurrentPlayerType = PlayerType.witch;
166                         player.Room = enemyani5.Room = CurrentRoom;
167                         if (Entered) { CurrentRoom.PlayerStartingPos
                        (witchplayerani, 40, 480); }
168                         break;
169                     case RoomType.Winner:
170                         CurrentRoom = Winner;
171                         CurrentPlayerType = PlayerType.witch;
172                         player.Room = CurrentRoom;
173                         if (Entered) { CurrentRoom.PlayerStartingPos
                        (witchplayerani, 100, 500); }
174                         break;
175                 }
176                 CurrentRoom.Draw();
177 
178                 // Draws current player type
179                 if (CurrentPlayerType == PlayerType.cartoon)
180                 {
181                     player = cartoonplayerani;
182                     cartoonplayerani.Draw();
183                     cartoonplayerani.Update();
184                 }
185                 if (CurrentPlayerType == PlayerType.witch &&
                    ((CurrentRoomType == RoomType.BA || CurrentRoomType ==
                    RoomType.ATC || CurrentRoomType == RoomType.Library ||
                    CurrentRoomType == RoomType.Dungeon )))
186                 {
187                         foreach (var enemy in enemies)
188                         {
189                         if (enemy.Room == CurrentRoom)
190                         {
191                             enemy.Draw();
192                             enemy.Update();
193                             player1.Y = enemy.Y + 5;
194                             //animation of enemy changes to death when
                        distance between enemy and player are close and skill is
                         used by pressing mouse left click
195                             if (player1.PlayerIsAt(player.X, player.Y +
                        20, enemy.X, enemy.Y) && SplashKit.MouseClicked
                        (MouseButton.LeftButton) && (enemy.X - player.X <= 170)
                        && (player.Y - enemy.Y <= 30) && !enemy.Dead)
196                             {
197                                 enemy.EnemyWalkAni.Assign("noanimation2");
198                                 enemy.EnemyDieAni.Assign("dies");
199                                 enemy.Dead = true;
200                                 score += 10;
```

```
201                             }
202                             // player is dead if enemy touches player and
                    player respawns
203                             else if (player1.PlayerIsAt(player.X, player.Y
                     + 20, enemy.X, enemy.Y) && (enemy.X - player.X <= 20)
                    && (player.Y - enemy.Y <= 30))
204                             {
205                                 for (int i = 0; i < 1000000000; i++)
206                                 {
207                                     i += 1;
208                                 }
209                                 if (CurrentRoomType == RoomType.ATC)
210                                 {
211                                     CurrentRoom.PlayerStartingPos
                    (witchplayerani, 100, 530);
212                                 }
213                                 if (CurrentRoomType == RoomType.BA)
214                                 {
215                                     CurrentRoom.PlayerStartingPos
                    (witchplayerani, 20, 420);
216                                 }
217                                 if (CurrentRoomType == RoomType.Library)
218                                 {
219                                     CurrentRoom.PlayerStartingPos
                    (witchplayerani, 29, 363);
220                                 }
221                                 if (CurrentRoomType == RoomType.Dungeon)
222                                 {
223                                     CurrentRoom.PlayerStartingPos
                    (witchplayerani, 40, 480);
224                                 }
225                                 score = 0;
226                             }
227                         }
228                         }
229                      player = witchplayerani;
230                      witchplayerani.Draw();
231                      witchplayerani.Update();
232                 }
233
234                 SplashKit.DrawText(ScoreText, Color.Gold, font, 25 , 430,
                      10);
235
236                 window.Refresh(60);
237
238                 // coordinates for debugging purposes
239
240                 //Console.WriteLine($"X: {player1.X} Y: {player1.Y}");
241                 //Console.WriteLine($"enemy X: {enemyani.X} enemy Y:
```

```
                    {enemyani.Y}");
242                 //Console.WriteLine($"enemy2 X: {enemyani2.X} enemy2 Y:
                    {enemyani2.Y}");
243                 //Console.WriteLine($"player X: {player.X} player Y:
                    {player.Y} \n {enemyani.X - player.X}\n");
244                 //  Console.WriteLine($"X: {SplashKit.MousePosition().X}
                    Y: {SplashKit.MousePosition().Y}");
245             }
246             while (!window.CloseRequested);
247         }
248     }
249 }
```

```
1  using SplashKitSDK;
2
3  namespace NitsMercernary
4  {
5      public abstract class Room : Obj
6      {
7          private string _RoomName;
8          public Room(string[]ids,string name) :base(ids, name)
9          {
10             _RoomName = name;
11         }
12         public abstract void Draw();
13
14         public abstract void PlayerStartingPos(Player player,int x, int y);
15
16         public string RoomName { get { return _RoomName; } }
17     }
18 }
19
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace NitsMercernary
9  {
10     public class Winner : Room
11     {
12         private Bitmap WinnerRoom;
13         private int x, y;
14         public Winner(string[] ids, string name) : base(ids, name)
15         {
16             WinnerRoom = SplashKit.LoadBitmap("Winner", "images/
                 winningscreen.jpg");
17         }
18
19         public override void Draw()
20         {
21             WinnerRoom.Draw(0, 0);
22         }
23
24         public override void PlayerStartingPos(Player player, int x, int y)
25         {
26             player.X = x;
27             player.Y = y;
28         }
29         public int X { get { return x; } set { x = value; } }
30         public int Y { get { return y; } set { y = value; } }
31     }
32 }
33
```

```
 1  using SplashKitSDK;
 2  using System.Numerics;
 3
 4  namespace NitsMercernary
 5  {
 6      public class WitchPlayerAnimation : Player
 7      {
 8          public PlayerInterface Interface;
 9          public Bitmap witch;
10          public Bitmap witchATTACK;
11          public Bitmap witchwalk;
12          public Animation WitchIdleAni;
13          public Animation WitchAttackAni;
14          public Animation WitchWalkAni;
15          DrawingOptions WitchIdleOpt;
16          DrawingOptions WitchAttackOpt;
17          DrawingOptions WitchWalkOpt;
18          public bool idle;
19          public int i;
20          private int x, y;
21          public WitchPlayerAnimation(string name, string desc, int X, int
              Y) : base(name, desc, X, Y)
22          {
23              witch = SplashKit.LoadBitmap("witch", "images/
                  B_witch_idle.png");
24              witchATTACK = SplashKit.LoadBitmap("witchatk", "images/
                  B_witch_attack.png");
25              witchwalk = SplashKit.LoadBitmap("witchwalk", "images/
                  witch_walk.png");
26
27              witch.SetCellDetails(64, 96, 1, 6, 6);
28              witchATTACK.SetCellDetails(208, 92, 1, 9, 9);
29              witchwalk.SetCellDetails(64, 96, 2, 8, 16);
30
31              AnimationScript WitchIdleScript =
                  SplashKit.LoadAnimationScript("witchidle", "witchidle.txt");
32              AnimationScript WitchAttackScript =
                  SplashKit.LoadAnimationScript("witchattack",
                  "witchattack.txt");
33              AnimationScript WitchWalkScript =
                  SplashKit.LoadAnimationScript("witchwalk", "witchwalk.txt");
34
35              WitchIdleAni = WitchIdleScript.CreateAnimation("idle");
36              WitchAttackAni = WitchAttackScript.CreateAnimation("noanim");
37              WitchWalkAni = WitchWalkScript.CreateAnimation("noanima");
38
39              i = 0;
40              idle = true;
41
```

```
42              x = X;
43              y = Y;
44          }
45          public override void Update()
46          {
47
48              if (!idle)
49              {
50                  i += 1;
51              }
52
53              if ((!idle) && (i % 80 == 0))
54              {
55                  idle = true;
56                  WitchIdleAni.Assign("idle");
57                  Console.WriteLine("idle");
58                  WitchAttackAni.Assign("noanim");
59              }
60              if (SplashKit.MouseClicked(MouseButton.LeftButton))
61              {
62                  Console.WriteLine("atk");
63                  WitchIdleAni.Assign("noani");
64                  WitchAttackAni.Assign("attack");
65                  idle = false;
66              }
67              if ((SplashKit.KeyTyped(KeyCode.RightKey)) ||                ↵
                  (SplashKit.KeyTyped(KeyCode.DKey)))
68              {
69
70                  Console.WriteLine("walkright");
71                  WitchIdleAni.Assign("noani");
72                  WitchAttackAni.Assign("noanim");
73                  WitchWalkAni.Assign("walkright");
74
75              }
76              if ((SplashKit.KeyReleased(KeyCode.RightKey)) ||             ↵
                  (SplashKit.KeyReleased(KeyCode.DKey)))
77              {
78                  WitchIdleAni.Assign("idle");
79                  WitchWalkAni.Assign("noanima");
80              }
81              if ((SplashKit.KeyTyped(KeyCode.LeftKey)) ||                 ↵
                  (SplashKit.KeyTyped(KeyCode.AKey)))
82              {
83                  Console.WriteLine("walkleft");
84                  WitchIdleAni.Assign("noani");
85                  WitchAttackAni.Assign("noanim");
86                  WitchWalkAni.Assign("walkleft");
87              }
```

```
88              if ((SplashKit.KeyReleased(KeyCode.LeftKey)) ||
                  (SplashKit.KeyReleased(KeyCode.AKey)))
89              {
90                  WitchIdleAni.Assign("idle");
91                  WitchWalkAni.Assign("noanima");
92              }
93              PlayerMovement();
94              WitchIdleAni.Update();
95              WitchAttackAni.Update();
96              WitchWalkAni.Update();
97          }
98
99          public void PlayerMovement()
100         {
101             if ((SplashKit.KeyDown(KeyCode.RightKey)) ||
                  (SplashKit.KeyDown(KeyCode.DKey)))
102             {
103                 if (X <= 550)
104                 { X += 3; }
105                 else { X = 555; }
106             }
107             if ((SplashKit.KeyDown(KeyCode.LeftKey)) || (SplashKit.KeyDown
                  (KeyCode.AKey)))
108             {
109                 if (X >= 5)
110                 { X -= 3; }
111                 else { X = 0; }
112             }
113
114         }
115         public override void Draw()
116         {
117             WitchIdleOpt = SplashKit.OptionWithAnimation(WitchIdleAni);
118             WitchAttackOpt = SplashKit.OptionWithAnimation
                  (WitchAttackAni);
119             WitchWalkOpt = SplashKit.OptionWithAnimation(WitchWalkAni);
120
121             SplashKit.DrawBitmap(witch, X - 20, Y - 40, WitchIdleOpt);
122             SplashKit.DrawBitmap(witchATTACK, X - 10, Y - 35,
                  WitchAttackOpt);
123             SplashKit.DrawBitmap(witchwalk, X - 10, Y - 35, WitchWalkOpt);
124         }
125
126         public override int X { get { return x; } set { x = value; } }
127         public override int Y { get { return y; } set { y = value; } }
128
129     }
130 }
```

```
1  using SplashKitSDK;
2
3  namespace NitsMercernary
4  {
5      public class ATC : Room
6      {
7          private int x, y;
8          private Bitmap _ATC_Building_In;
9          public ATC(string[]ids ,string name) : base(ids,name)
10         {
11             _ATC_Building_In = SplashKit.LoadBitmap("ATC_In", "images/
                 ATC_Building_In.png");
12         }
13
14         public override void Draw()
15         {
16             _ATC_Building_In.Draw(0, 0);
17         }
18
19         public override void PlayerStartingPos(Player player, int x, int y)
20         {
21             player.X = x;
22             player.Y = y;
23         }
24         public int X { get { return x; } set { x = value; } }
25         public int Y { get { return y; } set { y = value; } }
26     }
27 }
28
```

```
1  using SplashKitSDK;
2
3  namespace NitsMercernary
4  {
5      public class BA : Room
6      {
7          private Bitmap _BA_Building_In;
8          public BA(string[] ids, string name):base(ids, name)
9          {
10             _BA_Building_In = SplashKit.LoadBitmap("BA_In", "images/
                 BA_Building_inside.png");
11
12         }
13
14         public override void Draw()
15         {
16             _BA_Building_In.Draw(0, 0);
17         }
18
19         public override void PlayerStartingPos(Player player, int x, int y)
20         {
21             player.X = x;
22             player.Y = y;
23         }
24
25     }
26 }
27
```

```
 1  using SplashKitSDK;
 2
 3  namespace NitsMercernary
 4  {
 5      public class CartoonPlayerAnimation : Player
 6      {
 7          public Bitmap cartoon;
 8          public Animation test;
 9          public DrawingOptions opt;
10          private int x;
11          private int y;
12
13          public CartoonPlayerAnimation(string name, string desc, int X, int
               Y) : base(name, desc, X, Y)
14          {
15              cartoon = SplashKit.LoadBitmap("cartoon", "images/
                  character.png");
16              cartoon.SetCellDetails(48, 48, 4, 4, 16);
17              AnimationScript Walk = SplashKit.LoadAnimationScript("",
                  "test.txt");
18              test = Walk.CreateAnimation("walkback");
19              x = X;
20              y = Y;
21          }
22          public override void Update()
23          {
24
25              if (SplashKit.KeyDown(KeyCode.PKey))
26              { test.Assign("Dance"); }
27              if ((SplashKit.KeyTyped(KeyCode.UpKey)) || (SplashKit.KeyTyped
                  (KeyCode.WKey)))
28              {
29                  test.Assign("GoWalkBack");
30                  Console.WriteLine(test);
31              }
32              if ((SplashKit.KeyReleased(KeyCode.UpKey)) ||
                  (SplashKit.KeyReleased(KeyCode.WKey)))
33              {
34                  test.Assign("WalkBack");
35              }
36              if ((SplashKit.KeyTyped(KeyCode.DownKey)) ||
                  (SplashKit.KeyTyped(KeyCode.SKey)))
37              {
38                  test.Assign("GoWalkFront");
39              }
40              if ((SplashKit.KeyReleased(KeyCode.DownKey)) ||
                  (SplashKit.KeyReleased(KeyCode.SKey)))
41              {
42                  test.Assign("WalkFront");
```

```
43              }
44              if ((SplashKit.KeyTyped(KeyCode.RightKey)) ||
                  (SplashKit.KeyTyped(KeyCode.DKey)))
45              {
46                  { test.Assign("GoWalkRight"); }
47              }
48              if ((SplashKit.KeyReleased(KeyCode.RightKey)) ||
                  (SplashKit.KeyReleased(KeyCode.DKey)))
49              {
50                  { test.Assign("WalkRight"); }
51              }
52              if ((SplashKit.KeyTyped(KeyCode.LeftKey)) ||
                  (SplashKit.KeyTyped(KeyCode.AKey)))
53              {
54                  { test.Assign("GoWalkLeft"); }
55              }
56              if ((SplashKit.KeyReleased(KeyCode.LeftKey)) ||
                  (SplashKit.KeyReleased(KeyCode.AKey)))
57              {
58                  { test.Assign("WalkLeft"); }
59              }
60              PlayerMovement();
61              test?.Update();
62          }
63
64          public void PlayerMovement()
65          {
66              if ((SplashKit.KeyDown(KeyCode.UpKey)) || (SplashKit.KeyDown
                  (KeyCode.WKey)))
67              {
68                  if (Y >= 3)
69                  { Y -= 3; }
70                  else { Y = 0; }
71              }
72              if ((SplashKit.KeyDown(KeyCode.DownKey)) || (SplashKit.KeyDown
                  (KeyCode.SKey)))
73              {
74                  if (Y <= 550)
75                  { Y += 3; }
76                  else { Y = 555; }
77              }
78              if ((SplashKit.KeyDown(KeyCode.RightKey)) ||
                  (SplashKit.KeyDown(KeyCode.DKey)))
79              {
80                  if (X <= 550)
81                  { X += 3; }
82                  else { X = 555; }
83              }
84              if ((SplashKit.KeyDown(KeyCode.LeftKey)) || (SplashKit.KeyDown
```

```
                (KeyCode.AKey)))
 85             {
 86                 if (X >= 5)
 87                 { X -= 3; }
 88                 else { X = 0; }
 89             }
 90         }
 91
 92         public override void Draw()
 93         {
 94             opt = SplashKit.OptionWithAnimation(test);
 95             SplashKit.DrawBitmap(cartoon, X, Y, opt);
 96         }
 97
 98         public override int X { get { return x; } set { x = value; } }
 99         public override int Y { get { return y; } set { y = value; } }
100     }
101 }
102
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace NitsMercernary
9  {
10     public class Dungeon : Room
11     {
12         private Bitmap DungeonRoom;
13         private int x, y;
14         public Dungeon(string[] ids, string name) : base(ids, name)
15         {
16             DungeonRoom = SplashKit.LoadBitmap("DungeonIn", "images/
                 DungeonIn.jpg");
17         }
18
19         public override void Draw()
20         {
21             DungeonRoom.Draw(0, 0);
22         }
23
24         public override void PlayerStartingPos(Player player, int x, int y)
25         {
26             player.X = x;
27             player.Y = y;
28         }
29         public int X { get { return x; } set { x = value; } }
30         public int Y { get { return y; } set { y = value; } }
31     }
32 }
33
```

```
 1  using SplashKitSDK;
 2  using System;
 3  using System.Collections.Generic;
 4  using System.Linq;
 5  using System.Text;
 6  using System.Threading.Tasks;
 7
 8  namespace NitsMercernary
 9  {
10      public class EnemyAnimation : Player
11      {
12          public Bitmap Enemy;
13          public Bitmap EnemyAttack;
14          public Bitmap EnemyWalk;
15          public Bitmap EnemyDie;
16
17          public Animation EnemyIdleAni;
18          public Animation EnemyAttackAni;
19          public Animation EnemyWalkAni;
20          public Animation EnemyDieAni;
21
22          DrawingOptions EnemyIdleOpt;
23          DrawingOptions EnemyAttackOpt;
24          DrawingOptions EnemyWalkOpt;
25          DrawingOptions EnemyDieOpt;
26
27          public bool respawn;
28          public int i;
29          private int x, y;
30
31          EnemyDirection enemyDirection = EnemyDirection.Left;
32          private enum EnemyDirection
33          {
34              Left,
35              Right,
36              Dead
37          }
38          public EnemyAnimation(string name, string desc, int X, int Y) :
              base(name, desc, X, Y)
39          {
40              Enemy = SplashKit.LoadBitmap("enemy", "images/EnemyIdle.png");
41              EnemyAttack = SplashKit.LoadBitmap("enemyattack", "images/
                  EnemyAttack.png");
42              EnemyWalk = SplashKit.LoadBitmap("enemywalk", "images/
                  EnemyRun.png");
43              EnemyDie = SplashKit.LoadBitmap("enemydie", "images/
                  EnemyDie.png");
44
45              Enemy.SetCellDetails(80, 64, 7, 1, 7);
```

```
46              EnemyAttack.SetCellDetails(80, 64, 10, 1, 10);
47              EnemyWalk.SetCellDetails(80, 64, 8, 2, 16);
48              EnemyDie.SetCellDetails(80, 64, 15, 1, 15);
49
50              AnimationScript EnemyIdleScript =
                  SplashKit.LoadAnimationScript("enemyidle", "enemyidle.txt");
51              AnimationScript EnemyAttackScript =
                  SplashKit.LoadAnimationScript("enemyattack",
                  "enemyattack.txt");
52              AnimationScript EnemyWalkScript =
                  SplashKit.LoadAnimationScript("enemywalk", "enemywalk.txt");
53              AnimationScript EnemyDieScript = SplashKit.LoadAnimationScript
                  ("enemydie", "enemydie.txt");
54
55              EnemyIdleAni = EnemyIdleScript.CreateAnimation("noanimation");
56              EnemyAttackAni = EnemyAttackScript.CreateAnimation
                  ("noanimation1");
57              EnemyWalkAni = EnemyWalkScript.CreateAnimation("walkleft1");
58              EnemyDieAni = EnemyDieScript.CreateAnimation("noanimation3");
59
60              i = 0;
61              respawn = false;
62              x = X;
63              y = Y;
64          }
65          public override void Update()
66          {
67              if (X < 295 && !Dead)
68              { enemyDirection = EnemyDirection.Right; X = 300;
                  EnemyWalkAni.Assign("walkright1"); }
69              else if (X > 550 && !Dead)
70              { enemyDirection = EnemyDirection.Left; X = 545;
                  EnemyWalkAni.Assign("walkleft1"); }
71              else if (Dead)
72              { enemyDirection = EnemyDirection.Dead; }
73
74              if (respawn)
75              {
76                  i += 1;
77              }
78
79              switch (enemyDirection)
80              {
81                  case EnemyDirection.Left:
82                      X -= 1;
83                      break;
84                  case EnemyDirection.Right:
85                      X += 1;
86                      break;
```

```
87                  case EnemyDirection.Dead:
88
89                      if (i % 1000 == 0)
90                      {
91                      EnemyDieAni.Assign("noanimation3");
92                      enemyDirection = EnemyDirection.Right;
                       EnemyWalkAni.Assign("walkright1");
93                      }
94                      respawn = true;
95
96                      Dead = false;
97                      break;
98              }
99              EnemyWalkAni.Update();
100             EnemyIdleAni.Update();
101             EnemyAttackAni.Update();
102             EnemyDieAni.Update();
103         }
104         public override void Draw()
105         {
106             EnemyIdleOpt = SplashKit.OptionWithAnimation(EnemyIdleAni);
107             EnemyAttackOpt = SplashKit.OptionWithAnimation
                (EnemyAttackAni);
108             EnemyWalkOpt = SplashKit.OptionWithAnimation(EnemyWalkAni);
109             EnemyDieOpt = SplashKit.OptionWithAnimation(EnemyDieAni);
110
111             SplashKit.DrawBitmap(Enemy, X - 20, Y - 40, EnemyIdleOpt);
112             SplashKit.DrawBitmap(EnemyAttack, X - 10, Y - 35,
                EnemyAttackOpt);
113             SplashKit.DrawBitmap(EnemyWalk, X - 10, Y - 35, EnemyWalkOpt);
114             SplashKit.DrawBitmap(EnemyDie, X - 10, Y - 35, EnemyDieOpt);
115         }
116
117         public override int X { get { return x; } set { x = value; } }
118         public override int Y { get { return y; } set { y = value; } }
119
120     }
121 }
122
```

```
1
2
3  namespace NitsMercernary
4  {
5      public class IDgame
6      {
7          public List<string> _identifiers = new();
8
9          public IDgame(string[] idents)
10         {
11             _identifiers.AddRange(idents);
12         }
13
14         public bool AreYou(string id)
15         {
16             foreach (string ident in _identifiers)
17             {
18                 if (ident.ToLower() == id.ToLower()) { return true; }
19             }
20             return false;
21         }
22
23         public void AddIdentifier(string id)
24         {
25             _identifiers.Add(id.ToLower());
26         }
27
28     }
29 }
30
```

```
1  using SplashKitSDK;
2
3  namespace NitsMercernary
4  {
5      public class JohnStreet:Room
6      {
7          private string name;
8          private Bitmap _bitmap;
9          private Bitmap _BA_Building;
10         private Bitmap _Library;
11         private Bitmap _ATC_Building;
12         private Bitmap DungeonRoom;
13         public JohnStreet(string[]ids,string Name) : base(ids, Name)
14         {
15             name = Name;
16             _bitmap = SplashKit.LoadBitmap("grass", "images/
                 JohnStreet.png");
17             _Library = SplashKit.LoadBitmap("Library", "images/
                 Library.png");
18             _BA_Building = SplashKit.LoadBitmap("BA", "images/
                 BA_Building.png");
19             _ATC_Building = SplashKit.LoadBitmap("ATC", "images/
                 ATC_Building.png");
20             DungeonRoom = SplashKit.LoadBitmap("Dungeon", "images/
                 building.png");
21         }
22
23         public override void Draw()
24         {
25             _bitmap.Draw(0, 0);
26             _Library.Draw(260, -10);
27             _BA_Building.Draw(-20, 150);
28             _ATC_Building.Draw(450, 100);
29             DungeonRoom.Draw(355, 330);
30         }
31
32         public override void PlayerStartingPos(Player player, int x, int y)
33         {
34             player.X = x;
35             player.Y = y;
36
37         }
38
39         public string Name { get { return name; } }
40     }
41 }
42
```

```
1  using SplashKitSDK;
2
3  namespace NitsMercernary
4  {
5      public class Library : Room
6      {
7          private Bitmap _Library_Building_In;
8          private int x, y;
9          public Library(string[] ids, string name):base(ids, name)
10         {
11             _Library_Building_In = SplashKit.LoadBitmap("Library_In",
                 "images/Library.jpg");
12         }
13
14         public override void Draw()
15         {
16             _Library_Building_In.Draw(0,0);
17         }
18
19         public override void PlayerStartingPos(Player player,int x,int y)
20         {
21             player.X = x;
22             player.Y = y;
23         }
24         public int X { get { return x; } set { x = value; } }
25         public int Y { get { return y; } set { y = value; } }
26     }
27 }
28
```

```
 1
 2
 3  namespace NitsMercernary
 4  {
 5      public abstract class Obj : IDgame
 6      {
 7          private string _description;
 8          private string _name;
 9          public Obj(string[] ids,string name) : base(ids)
10          {
11              _name = name;
12          }
13
14          public string Name {  get { return _name; } }
15
16          public virtual string FullDescription {  get { return
              _description;} }
17      }
18  }
19
```

```
1  using SplashKitSDK;
2  using System.Transactions;
3
4  namespace NitsMercernary
5  {
6      public class Player : Obj, PlayerInterface
7      {
8          private int x;
9          private int y;
10
11         private bool dead;
12
13         private Room room;
14         public Player(string name, string desc, int X, int Y) : base(new
             string[] { "player" }, name)
15         {
16             x = X;
17             y = Y;
18             dead = false;
19             room = new JohnStreet(new string[] { "JohnStreet" },
                 "JohnStreet");
20         }
21         public virtual void Draw()
22         {
23         }
24
25         public virtual void Update()
26         {
27         }
28
29         public bool PlayerIsAt(int x1, int y1, int x2, int y2)
30         {
31             if ((x1 <= X) && (x2 >= X))
32             {
33                 if ((y1 >= Y) && (y2 <= Y)) { return true; }
34             }
35             return false;
36         }
37         public override string FullDescription { get { return $"{Name},
             {base.FullDescription}\nYou are carrying:\n"; } }
38
39         public virtual int X { get { return x; } set { x = value; } }
40         public virtual int Y { get { return y; } set { y = value; } }
41
42         public Room Room
43         {
44             get { return room; }
45             set { room = value; }
46         }
```

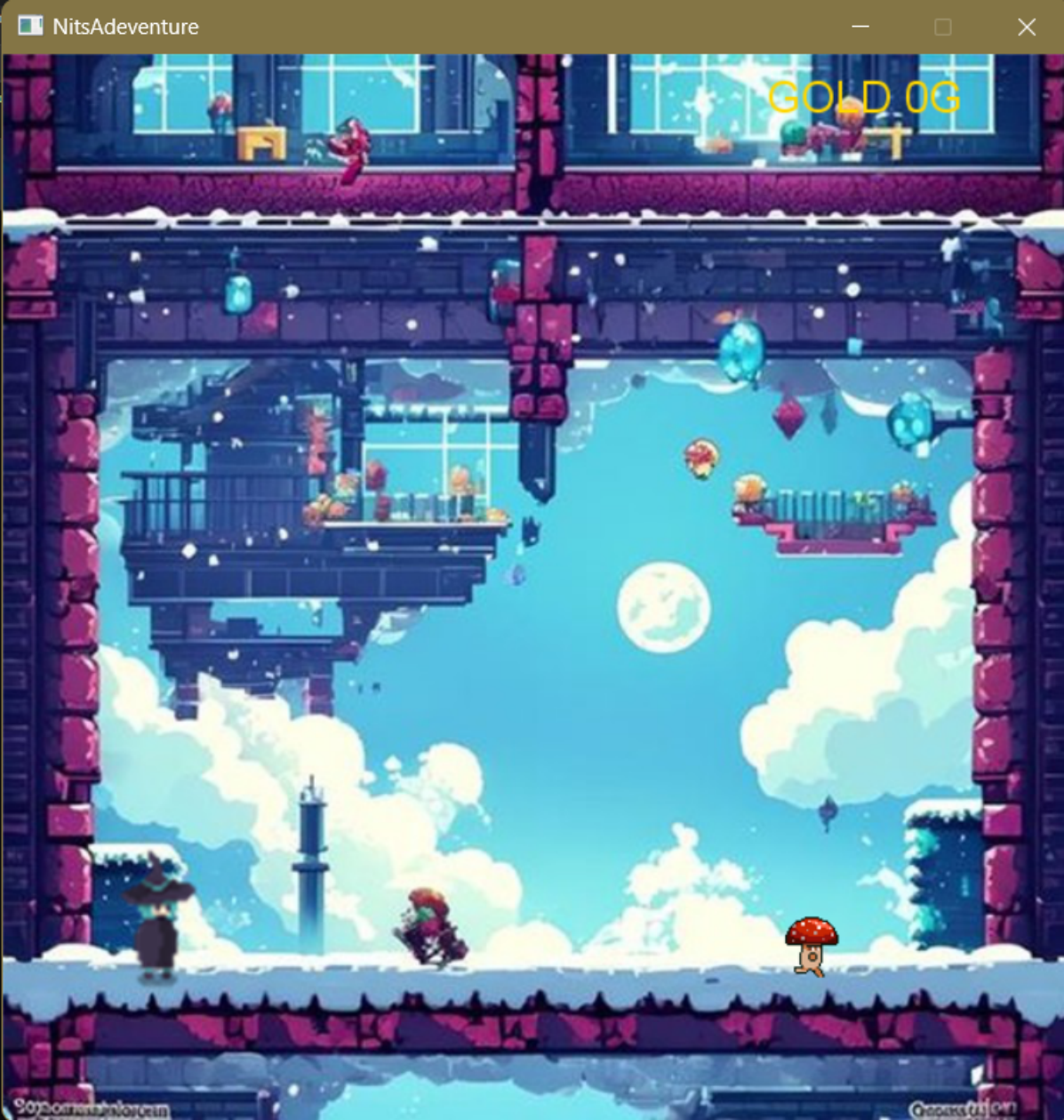
```
47
48          public bool Dead { get { return dead; } set { dead = value; } }
49      }
50  }
51
```

```
1
2  namespace NitsMercernary
3  {
4      public interface PlayerInterface
5      {
6          public bool PlayerIsAt(int x1, int y1, int x2, int y2);
7      }
8  }
9
```


NitsAdeventure
GOLD 100G
YOU WIN!


NitsAdeventure
Inside ATC Building
GOLD 0G


NitsAdeventure
BA Building
GOLD 20G


NitsAdeventure
GOLD 0G


NitsAdeventure
Library
GOLD 0G
ATC Buildin
BA Building