9/26/2018

Updates

GADE6112 – Task 3

Ashveer Jugdav - 17611612
LECTURER: KRUBEN NAIDOO

GADE6112 – Task 3
Updates.docx
Ashveer Jugdav - 17611612

The ability for units to change team allegiance after a certain in-game event (10 marks)

I used a button to simulate the in-game event, teams will change allegiance as soon as the user pushes the button included in the User Interface. The button will also "grey out" so the user cannot press it again.

Code in Form1 for the button:

```
private void btninGameEvent_Click(object sender, EventArgs e)
{
    ge.GEinGameEvent();
    btninGameEvent.Enabled = false;
}
```

Code for the actual In Game Event:

```
public void inGameEvent()
{
    foreach (Unit u in unitsOnMap)
    {
        if (u.FactionTeam == "Red" && u.SymbolImage == "M")
        {
            u.FactionTeam = "Blue";
            u.SymbolImage = "m";
            map[u.XPosition, u.YPosition] = u.SymbolImage;
        }
        else if (u.FactionTeam == "Red" && u.SymbolImage == "R")
        {
            u.FactionTeam = "Blue";
            u.SymbolImage = "r";
            map[u.XPosition, u.YPosition] = u.SymbolImage;
        }
        else if (u.FactionTeam == "Blue" && u.SymbolImage == "m")
        {
            u.FactionTeam = "Red";
            u.SymbolImage = "M";
            map[u.XPosition, u.YPosition] = u.SymbolImage;
        }
        else if (u.FactionTeam == "Blue" && u.SymbolImage == "r")
        {
            u.FactionTeam = "Red";
            u.SymbolImage = "R";
            map[u.XPosition, u.YPosition] = u.SymbolImage;
        }
    }

    foreach (Building b in buildingsOnMap)
    {
        if (b.FactionTeam == "Red" && b.SymbolImage == "$")
        {
            Debug.WriteLine("BUILDING MOVE1");
            b.FactionTeam = "Blue";
            b.SymbolImage = "@";
            map[b.XPosition, b.YPosition] = b.SymbolImage;
        }
        else if (b.FactionTeam == "Blue" && b.SymbolImage == "@")
        {
            Debug.WriteLine("BUILDING MOVE2");
            b.FactionTeam = "Red";
            b.SymbolImage = "$";
            map[b.XPosition, b.YPosition] = b.SymbolImage;
        }
    }
}
```

GADE6112 – Task 3
Updates.docx
Ashveer Jugdav - 17611612

The introduction of "neutral" enemies that both teams need to fight. Neutral enemies attack their closest enemy, regardless of team, but do not attack buildings (5 marks)

I created a new class called NeutralUnit and used the same base code for the Melee and Ranged Units but changed their attack range to 3 [because they are meant to be pirates and can go "off the book" shall we say] and included a special rule set for this unit. This unit is spawned regardless whether the other two have been spawned.

```
public void battlefield()  //A method to randomly generate a new battle field with random units.
{
    Random rnd = new Random();

    if (map[buildingsOnMap[0].XPosition - 1, buildingsOnMap[0].YPosition] == "." && map[buildingsOnMap[1].XPosition + 1, buildingsOnMap[1].YPosition] == ".")
    {
        int unitSelection = rnd.Next(1, 100);
        int teamSelection = rnd.Next(1, 100);

        if ((unitSelection) % 2 == 0)
        {
            if ((teamSelection) % 2 == 0)
            {
                Unit tmp = new MeleeUnit(buildingsOnMap[0].XPosition - 1, buildingsOnMap[0].YPosition, 100, 1, false, 1, "Red", "M", "Joe");
                map[tmp.XPosition, tmp.YPosition] = tmp.SymbolImage;
                unitsOnMap.Add(tmp);
            }
            else
            {
                Unit tmp = new MeleeUnit(buildingsOnMap[1].XPosition + 1, buildingsOnMap[1].YPosition, 100, 1, false, 1, "Blue", "m", "Connor");
                map[tmp.XPosition, tmp.YPosition] = tmp.SymbolImage;
            }
        }
        else
        {
            if ((teamSelection) % 2 == 0)
            {
                Unit tmp = new RangedUnit(buildingsOnMap[0].XPosition - 1, buildingsOnMap[0].YPosition, 100, 1, false, 2, "Red", "R", "shalom");
                map[tmp.XPosition, tmp.YPosition] = tmp.SymbolImage;
                unitsOnMap.Add(tmp);
            }
            else
            {
                Unit tmp = new RangedUnit(buildingsOnMap[1].XPosition + 1, buildingsOnMap[1].YPosition, 100, 1, false, 2, "Blue", "r", "bob");
                map[tmp.XPosition, tmp.YPosition] = tmp.SymbolImage;
                unitsOnMap.Add(tmp);
            }
        }

        Unit n = new NeutralUnit(rnd.Next(0, 19), rnd.Next(0, 19), 100, 1, false, 3, "Neutral", "N", "mushi mush");
        map[n.XPosition, n.YPosition] = n.SymbolImage;
        unitsOnMap.Add(n);
    }
}
```

Where the code will spawn a new Neutral unit

GADE6112 – Task 3
Updates.docx
Ashveer Jugdav - 17611612

New unit or building types for each team (5 marks per new unit type)

For this I decided to go with a new building type which will affect each team, the Gateway. What this building does is if a unit were to pass over this building, a section of code would activate which "teleports" that unit to a random point on the map and they will have to continue moving from this point. The class for the Gateway inherits all the base properties of the Building class.

```
public void constructBuildings()
{
    Random rnd = new Random();

    Building resouceBuilding = new ResourceBuilding(map.Length-2, map.Length - 2, 100, "Red", "$", "something", 10, 50);
    map[resouceBuilding.XPosition, resouceBuilding.YPosition] = resouceBuilding.SymbolImage;
    buildingsOnMap.Add(resouceBuilding);

    Building factoryBuilding = new FactoryBuilding(1, 1, 100, "Blue", "@", 55, 50, 50);
    map[factoryBuilding.XPosition, factoryBuilding.YPosition] = factoryBuilding.SymbolImage;
    buildingsOnMap.Add(factoryBuilding);

    Building gatewayBuilding = new GatewayBuilding(rnd.Next(0, map.Length), rnd.Next(0, 19), 100, "none", "%");
    map[gatewayBuilding.XPosition, gatewayBuilding.YPosition] = gatewayBuilding.SymbolImage;
    buildingsOnMap.Add(gatewayBuilding);
}
```

Where the code will spawn the Gateway at a random location

```
#region GateWay Rules
if ((m.unitsOnMap[i].XPosition == m.buildingsOnMap[2].XPosition) && (m.unitsOnMap[i].YPosition == m.buildingsOnMap[2].YPosition))
{
    Random rnd = new Random();
    m.unitsOnMap[i].XPosition = rnd.Next(0, 19);
    m.unitsOnMap[i].YPosition = rnd.Next(0, 19);
}
#endregion
```

Special code which will check whether the current unit is on the same position of the Gateway on will then teleport it to a random location.

GADE6112 – Task 3
Updates.docx
Ashveer Jugdav - 17611612

Custom map sizing (5 marks)

What I had to do here was create a method within the Map Class which required the input of two variables [`int userX, int userY`] this method was then passed through to the game engine and from there to the main Form. The user defined variables were then taken from the textboxes and passed through this method all the way back to the Map Class. Within the map class these two variables were assigned to [`public static int userDefinedX; public static int userDefinedY;`] and used throughout the rest of the code to define the map size.

Before any of this takes place, all buttons on the UI are disabled, the user must enter values and click the register button for all the UI elements to be useable.

Map Class

```
public static int userDefinedX;
public static int userDefinedY;

public void getMapSize(int userX, int userY)
{
    userDefinedX = userX;
    userDefinedY = userY;
}

public string [,] map = new string[userDefinedX, userDefinedY];
public List<Unit> unitsOnMap = new List<Unit>();
public List<Building> buildingsOnMap = new List<Building>();
```

Method which grabs the variables from From1

Applied to the Map Size

Form1

```
private void btnRegister_Click(object sender, EventArgs e)
{
    userDefinedX = int.Parse(txtX.Text);
    userDefinedY = int.Parse(txtY.Text);
    ge.GEgetMapSize(userDefinedX, userDefinedY);
    ge.GEinitialiseMap();
    pnlUserMapSize.Enabled = false;
    btnStart.Enabled = true;
    btnPause.Enabled = true;
    btninGameEvent.Enabled = true;
}
```

Button code for custom map Size

Method where we are passing the variables to the Map Class