# How to create a Chess App using Unity

Workshop 3

## Prerequisites

-Unity installed (preferably version 2019.3 or newer)
-An IDE (code editor) that is attached to Unity
    -When installing Unity also install Visual Studio with it for it to work automatically
-The completed version: [https://github.umn.edu/app-developers-club/Chess_App](https://github.umn.edu/app-developers-club/Chess_App)

## Let's Get Started

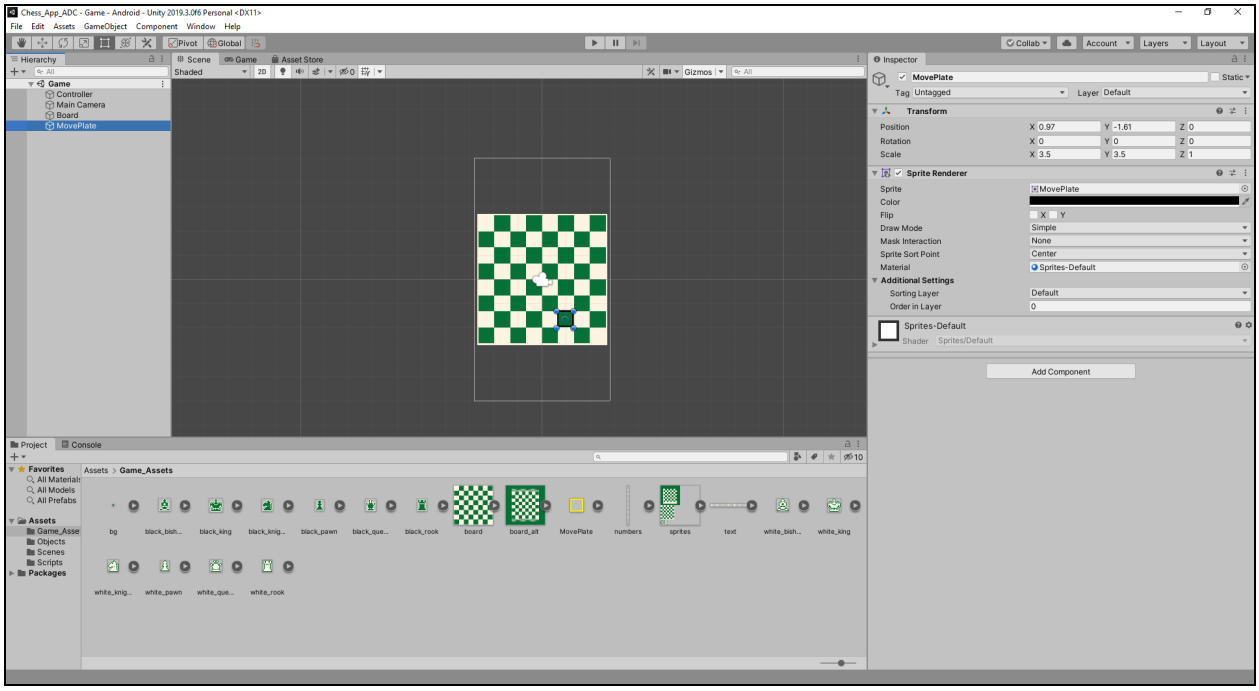
Adding position helper functions to the Game script:

- Open Unity and open the Game script
- This tutorial is going to deal a lot with creating movement functionality
    - It will be useful to have some general position functions
- Let's add three:
    - SetPositionEmpty : sets the board position empty
    - GetPosition: returns the GameObject that is on that position
    - PositionOnBoard: True or False depending on if the given position is even on the board

```
        //Overwrites either empty space or whatever was there
        positions[cm.GetXBoard(), cm.GetYBoard()] = obj;
    }

    public void SetPositionEmpty(int x, int y)
    {
        positions[x, y] = null;
    }

    public GameObject GetPosition(int x, int y)
    {
        return positions[x, y];
    }

    public bool PositionOnBoard(int x, int y)
    {
        if (x < 0 || y < 0 || x >= positions.GetLength(0) || y >= positions.GetLength(1)) return false;
        return true;
    }
}
```

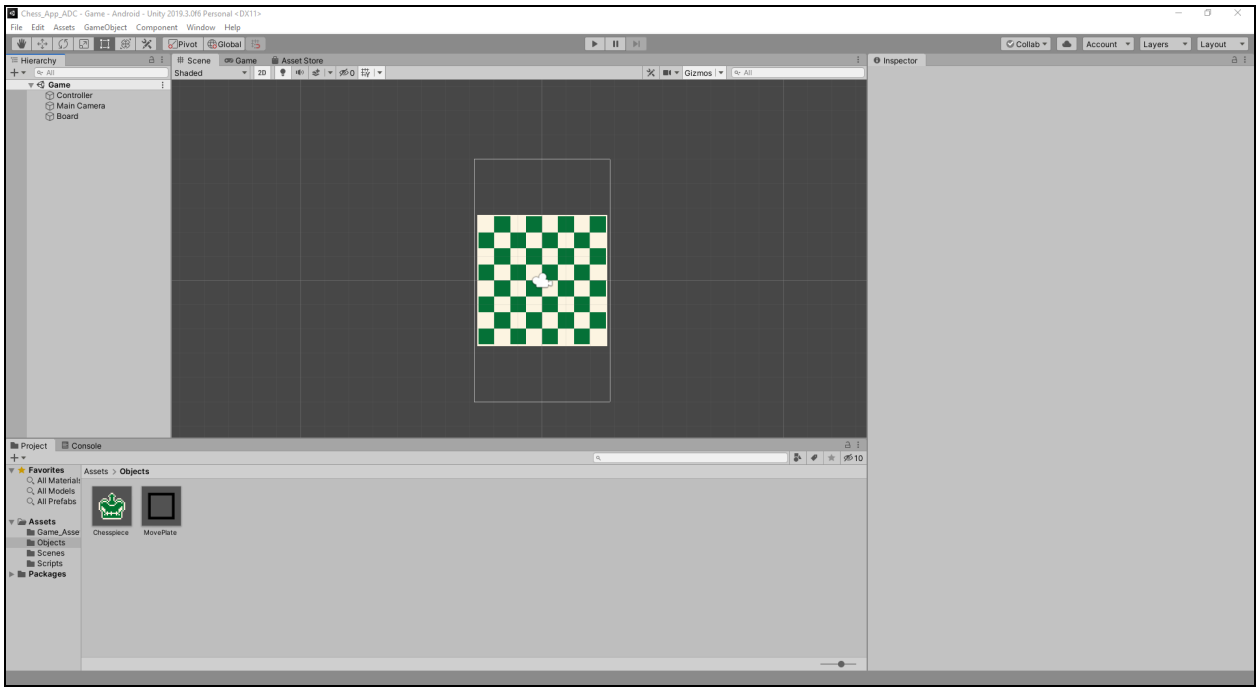
Adding MovePlate:

- MovePlate is another GameObject that we will need in order to have a ChessPiece move
- When a Chesspiece is selected we want to show all possible moves, so we will spawn a MovePlate at each location that the Chesspiece could move to
- The MovePlates will have a reference to the Chesspiece that spawned it so if you click on one of the MovePlates, it can correctly move the Chesspiece that it references to its location
- First we need to download the PNG file
    - https://github.umn.edu/app-developers-club/Chess_App/tree/master/TUTORIAL
    - Download ONLY MovePlate.png by clicking on it and then selecting download
- Once downloaded drag the file into the Sprites folder in Unity
- Select the Sprite and in the Inspector choose a Filter Mode of “Point (no filter)”
- Now drag it into the scene from the Game_Assets folder
    - Select apply changes
- Click on the MovePlate and set the X and Y scale to 3.5
- Change the color to Black




Making the MovePlate Object:

- Now select the Objects folder
- Drag the MovePlate from the top left pane into the Objects folder
- Delete the MovePlate from the top left pane




Creating MovePlate's Functionality:

- Now add a MovePlate script and open it
- There will be two types of MovePlates: movement, and attacking
- With that in mind let's add the basic attributes that this class needs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MovePlate : MonoBehaviour
{
    //Some functions will need reference to the controller
    public GameObject controller;

    //The Chesspiece that was tapped to create this MovePlate
    GameObject reference = null;

    //Location on the board
    int matrixX;
    int matrixY;

    //false: movement, true: attacking
    public bool attack = false;
}
```

Adding functions to the MovePlate script:

- There are many functions to implement
- Start: determine if the MovePlate is black or red
- OnMouseUp: When someone clicks on the MovePlate we want the Chesspiece to move to that location
    - This can be accomplished by updating it's XBoard and YBoard positions
    - Remember, after the position is updated the World Space position also needs to be updated and that can be done with Chessman.SetCoords()
- SetCoords: This is a new SetCoords for the MovePlate that can be used to set its position when it is spawned in
- SetReference: Also useful when creating a new Moveplate
- GetReference: A simple getter

```
//false: movement, true: attacking
public bool attack = false;

public void Start()
{
    if (attack)
    {
        //Set to red
        gameObject.GetComponent<SpriteRenderer>().color = new Color(1.0f, 0.0f, 0.0f, 1.0f);
    }
}

public void OnMouseUp()
{
    controller = GameObject.FindGameObjectWithTag("GameController");

    //Destroy the victim Chesspiece
    if (attack)
    {
        GameObject cp = controller.GetComponent<Game>().GetPosition(matrixX, matrixY);

        Destroy(cp);
    }
```

```
    //Set the Chesspiece's original location to be empty
    controller.GetComponent<Game>().SetPositionEmpty(reference.GetComponent<Chessman>().GetXBoard(),
        reference.GetComponent<Chessman>().GetYBoard());

    //Move reference chess piece to this position
    reference.GetComponent<Chessman>().SetXBoard(matrixX);
    reference.GetComponent<Chessman>().SetYBoard(matrixY);
    reference.GetComponent<Chessman>().SetCoords();

    //Update the matrix
    controller.GetComponent<Game>().SetPosition(reference);

    //Destroy the move plates including self
    reference.GetComponent<Chessman>().DestroyMovePlates();
}

public void SetCoords(int x, int y)
{
    matrixX = x;
    matrixY = y;
}

public void SetReference(GameObject obj)
{
    reference = obj;
```

```
    }

    public GameObject GetReference()
    {
        return reference;
    }
}
```

Now we need to switch over to the Chessman script:

- We can add all this code to the bottom of the script

```
private void OnMouseUp()
{
    //Remove all moveplates relating to previously selected piece
    DestroyMovePlates();

    //Create new MovePlates
    InitiateMovePlates();
}

public void DestroyMovePlates()
{
    //Destroy old MovePlates
    GameObject[] movePlates = GameObject.FindGameObjectsWithTag("MovePlate");
    for (int i = 0; i < movePlates.Length; i++)
    {
        Destroy(movePlates[i]); //Be careful with this function "Destroy" it is asynchronous
    }
  }

  public void InitiateMovePlates()
  {
    switch (this.name)
    {
        case "black_queen":
        case "white_queen":
            LineMovePlate(1, 0);
            LineMovePlate(0, 1);
            LineMovePlate(1, 1);
            LineMovePlate(-1, 0);
            LineMovePlate(0, -1);
```

```
            LineMovePlate(-1, -1);
            LineMovePlate(-1, 1);
            LineMovePlate(1, -1);
            break;
        case "black_knight":
        case "white_knight":
            LMovePlate();
            break;
        case "black_bishop":
        case "white_bishop":
            LineMovePlate(1, 1);
            LineMovePlate(1, -1);
            LineMovePlate(-1, 1);
            LineMovePlate(-1, -1);
            break;
        case "black_king":
        case "white_king":
            SurroundMovePlate();
            break;
        case "black_rook":
        case "white_rook":
            LineMovePlate(1, 0);
            LineMovePlate(0, 1);
            LineMovePlate(-1, 0);
            LineMovePlate(0, -1);
            break;
        case "black_pawn":
            PawnMovePlate(xBoard, yBoard - 1);
            break;
        case "white_pawn":
            PawnMovePlate(xBoard, yBoard + 1);
            break;
    }
}

public void LineMovePlate(int xIncrement, int yIncrement)
{
    Game sc = controller.GetComponent<Game>();

    int x = xBoard + xIncrement;
    int y = yBoard + yIncrement;

    while (sc.PositionOnBoard(x, y) && sc.GetPosition(x, y) == null)
    {
        MovePlateSpawn(x, y);
        x += xIncrement;
        y += yIncrement;
    }

    if (sc.PositionOnBoard(x, y) && sc.GetPosition(x, y).GetComponent<Chessman>().player != player)
    {
        MovePlateAttackSpawn(x, y);
    }
}

public void LMovePlate()
{
    PointMovePlate(xBoard + 1, yBoard + 2);
    PointMovePlate(xBoard - 1, yBoard + 2);
    PointMovePlate(xBoard + 2, yBoard + 1);
    PointMovePlate(xBoard + 2, yBoard - 1);
    PointMovePlate(xBoard + 1, yBoard - 2);
```

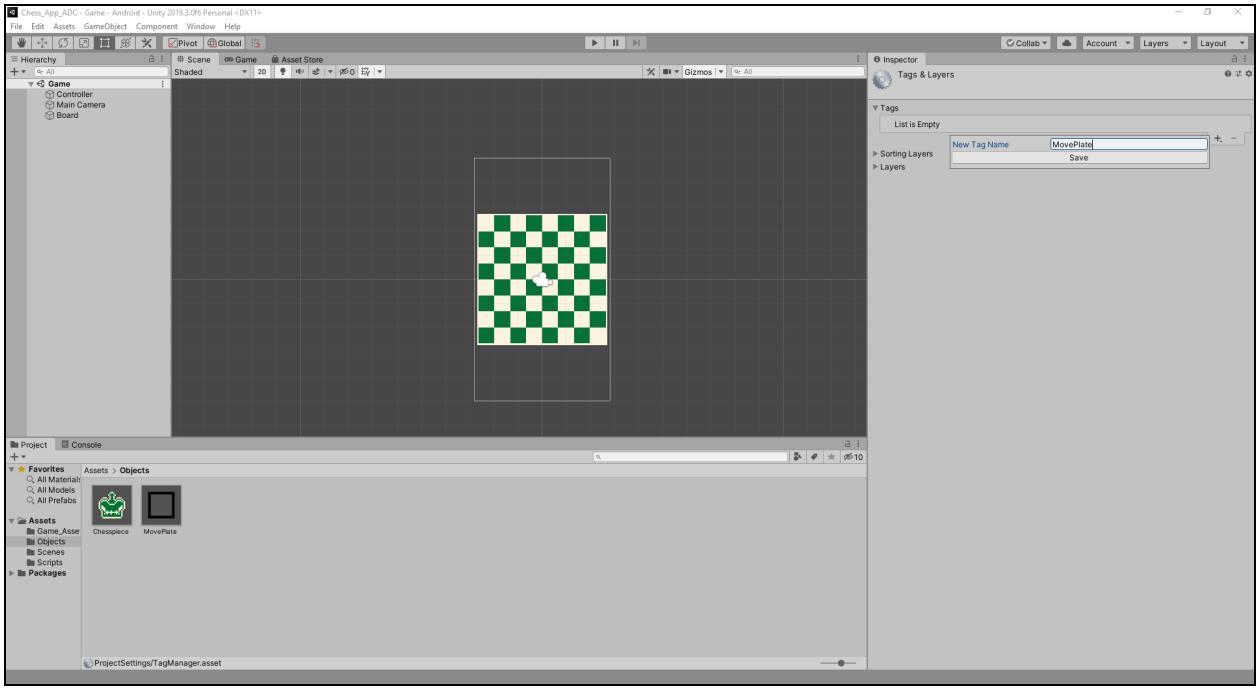
```
    PointMovePlate(xBoard - 1, yBoard - 2);
    PointMovePlate(xBoard - 2, yBoard + 1);
    PointMovePlate(xBoard - 2, yBoard - 1);
}

public void SurroundMovePlate()
{
    PointMovePlate(xBoard, yBoard + 1);
    PointMovePlate(xBoard, yBoard - 1);
    PointMovePlate(xBoard - 1, yBoard + 0);
    PointMovePlate(xBoard - 1, yBoard - 1);
    PointMovePlate(xBoard - 1, yBoard + 1);
    PointMovePlate(xBoard + 1, yBoard + 0);
    PointMovePlate(xBoard + 1, yBoard - 1);
    PointMovePlate(xBoard + 1, yBoard + 1);
}

public void PointMovePlate(int x, int y)
{
    Game sc = controller.GetComponent<Game>();
    if (sc.PositionOnBoard(x, y))
    {
        GameObject cp = sc.GetPosition(x, y);

        if (cp == null)
        {
            MovePlateSpawn(x, y);
        }
        else if (cp.GetComponent<Chessman>().player != player)
        {
            MovePlateAttackSpawn(x, y);
        }
    }
}

public void PawnMovePlate(int x, int y)
{
    Game sc = controller.GetComponent<Game>();
    if (sc.PositionOnBoard(x, y))
    {
        if (sc.GetPosition(x, y) == null)
        {
            MovePlateSpawn(x, y);
        }

        if (sc.PositionOnBoard(x + 1, y) && sc.GetPosition(x + 1, y) != null && sc.GetPosition(x + 1,
                y).GetComponent<Chessman>().player != player)
        {
            MovePlateAttackSpawn(x + 1, y);
        }

        if (sc.PositionOnBoard(x - 1, y) && sc.GetPosition(x - 1, y) != null && sc.GetPosition(x - 1,
                y).GetComponent<Chessman>().player != player)
        {
            MovePlateAttackSpawn(x - 1, y);
        }
    }
}

public void MovePlateSpawn(int matrixX, int matrixY)
{
    //Get the board value in order to convert to xy coords
```

```
        float x = matrixX;
        float y = matrixY;

        //Adjust by variable offset
        x *= 0.66f;
        y *= 0.66f;

        //Add constants (pos 0,0)
        x += -2.3f;
        y += -2.3f;

        //Set actual unity values
        GameObject mp = Instantiate(movePlate, new Vector3(x, y, -3.0f), Quaternion.identity);

        MovePlate mpScript = mp.GetComponent<MovePlate>();
        mpScript.SetReference(gameObject);
        mpScript.SetCoords(matrixX, matrixY);
    }

    public void MovePlateAttackSpawn(int matrixX, int matrixY)
    {
        //Get the board value in order to convert to xy coords
        float x = matrixX;
        float y = matrixY;

        //Adjust by variable offset
        x *= 0.66f;
        y *= 0.66f;

        //Add constants (pos 0,0)
        x += -2.3f;
        y += -2.3f;

        //Set actual unity values
        GameObject mp = Instantiate(movePlate, new Vector3(x, y, -3.0f), Quaternion.identity);

        MovePlate mpScript = mp.GetComponent<MovePlate>();
        mpScript.attack = true;
        mpScript.SetReference(gameObject);
        mpScript.SetCoords(matrixX, matrixY);
    }
```

Getting the MovePlates to work:

- Take a look at the code yourself, it does all the calculations needed for each object to spawn in the correct MovePlates
    - For example, it has a function to calculate an LMovePlate which is useful for a knight since it can move in an L
    - There is a switch statement that executes all the correct functions for the correct piece
- Now return to Unity after saving the scripts
- Select Chesspiece and let's drag MovePlate into the "Move Plate" box under the Chessman (Script) section
- Add a 2D box Collider as a component to the Chesspiece
    - Without a box collider an OnMouseUp() function would never run
    - Note: OnMouseUp() also works correctly when tapping a phone screen
- Select MovePlate and add the MovePlate script as a component, also add a 2D box collider

- Select MovePlate and under the Tag section add a new tag and call it “MovePlate”




Now if you press play you can actually play Chess:

- The basics of the game are almost complete