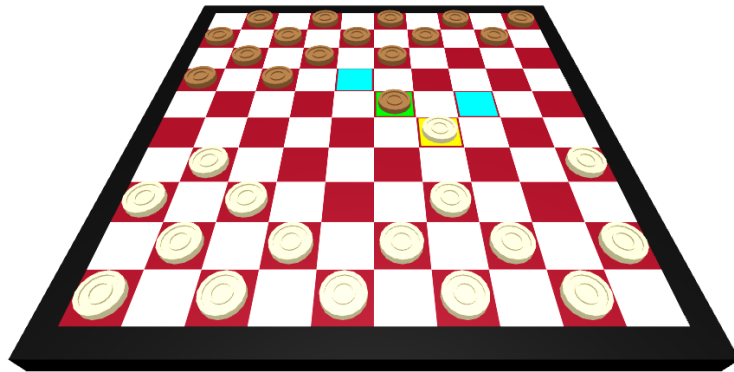


# Checkers Game

By WaveCave Games



# Description

This is a very simple double checkers game, which is played by 2 players. Eat all enemy's pieces to win the game. Include full source code so you can easily make an individual mode (as known as player vs. computer).

Features:

- Easy to win
- Ready to use 3D checkers pieces
- Include all basic piece movements (walk, eat, promotion)
- 'As much as possible' and 'one per moves' piece eating rule setting
- Open source
- Clean script interface

# How to Play

Open 'WaveCaveGames' -> 'CheckersGame' -> 'Scenes', open the 'Menu' scene, click 'Play' button.

In the settings, you can change piece eating rule (as much as possible, and one per move).

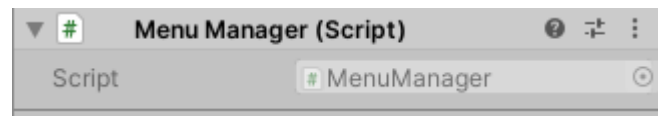
Eat all enemy's pieces to win the game.

For further help, see 'Help' in the game menu.

# Component Guide

## Menu Manager:

This script is used in button. It has no properties.



## Game Manager:

### Check Size:

The single check size. Chess board Surface size is check size \* 10.

### Check Trigger:

The check trigger. It automatically Create 50 triggers during Start() Function.

### Check Trigger Parent:

The parent object for all triggers.

### Piece Parent:

The parent object for all pieces.

### Piece Prefabs:

The prefab for all pieces. It automatically create 40 pieces during Start() function.

### King Piece Mesh:

The mesh for king piece.

### Piece Select Mark:

The mark you used for select a piece.

### Piece Move Mark:

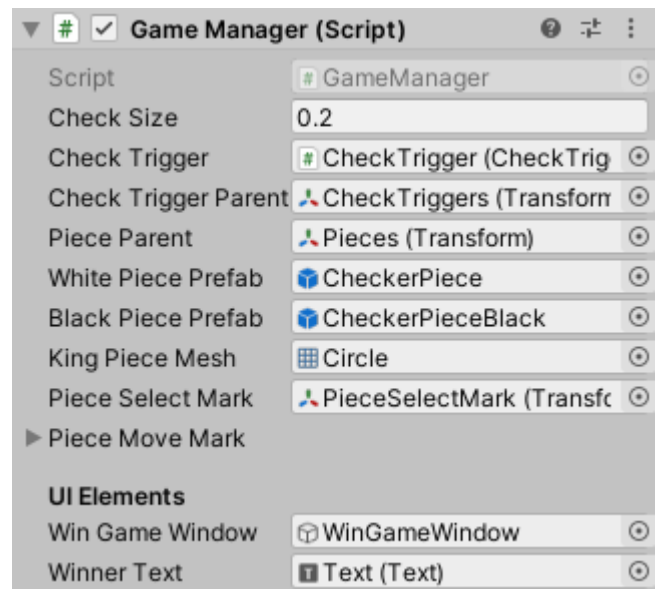
The mark you used after moving a piece. Size must be 2.

### Win Game Window:

The window displayed on UI. When you 'eat' all enemy's pieces, this window will appear.

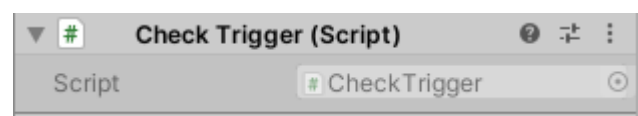
### Winner text:

The text inside the win game window, display whose win.



## Check Trigger:

When you clicked a piece, the trigger And its marks will appear. It has no Properties in inspector.



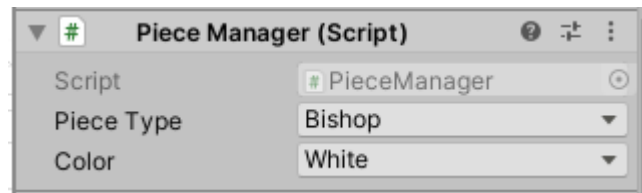
## Piece Manager:

### Piece Type:

The piece type (normal / king).

### Color:

The piece color (black / white).



# General Components

Selection:

**Buttons:**

All buttons inside this object.

**Key:**

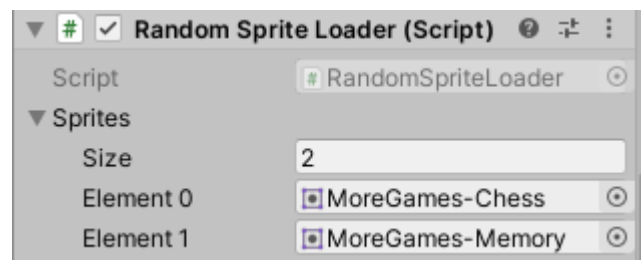
The key string you used for storing the data.



Random Sprite Loader:

**Sprites:**

Sprites you want to load randomly.



# Scripting

```
namespace WaveCaveGames.CheckersGame{

public class MenuManager : MonoBehaviour
{
public void LoadScene(string sceneName){}
public void Quit(){}
public void OpenURL(string url){}
}

public class GameManager : MonoBehaviour
{
public float checkSize;
public CheckTrigger checkTrigger;
public Transform checkTriggerParent;
public Transform pieceParent;
public GameObject whitePiecePrefab;
public GameObject blackPiecePrefab;
public Mesh kingPieceMesh;
public Transform pieceSelectMark;
public Transform[] pieceMoveMark;
[Header("UI Elements")]
public GameObject winGameWindow;
public Text winnerText;
//check trigger is laid out regularly, laid 5 horizontal checks out, then start laying next vertical checks. e.g. checkTriggers[0] position is (-4.5, -3.5), checkTriggers[6] position is (-3.5, -2.5), checkTriggers[25] position is (0.5, -4.5)
[HideInInspector] public CheckTrigger[] checkTriggers;
[HideInInspector] public PieceManager clickedPiece;
//true = black turn, false = white turn
[HideInInspector] public bool isBlackTurn;
[HideInInspector] public PieceManager[] whitePieces;
[HideInInspector] public PieceManager[] blackPieces;
public const float sqrt2 = 1.414214f;
public CheckTrigger FindCheck(Vector3 v){}
public void FindCheckAndLightUp(Vector3 v, bool eatPiece){}
//return check if can get to target, else return null. p: the selected piece. v: the check vector. eatPiece: if ticked, it will check if there is a piece can pass through.
public CheckTrigger FindAndReturnCheckIfCanGetToTarget(PieceManager p, Vector3 v, bool eatPiece){}
```

```

public void ClickPiece(PieceManager p){}
//this is same as ClickPiece(PieceManager), but only light up the
check that can eat piece, if can't eat piece, it will not light
up.
public void ClickPieceAgain(PieceManager p){}
//return all possible targets for piece p.
public CheckTrigger[] PossibleTargets(PieceManager p){}
//return all possible targets that can eat piece for piece p.
public CheckTrigger[] PossibleEatingTargets(PieceManager p){}
//check if can eat piece again, if can't eat piece, it will change
player turn.
public void CheckIfCanEatPieceAgain(PieceManager p){}
public void ChangePlayerTurn(){ }
public void CheckIfWin(){ }
public void BackToMenu(){ }
public CheckTrigger[] FindActiveTriggers(){ }
}

public class CheckTrigger : MonoBehaviour
{
[HideInInspector] public PieceManager piece;
[HideInInspector] public bool topPlace;
public void ClickThis(){ }
}

public enum PieceType{
Normal, King
}
public enum PieceColor{
White, Black
}
public class PieceManager : MonoBehaviour
{
public PieceType pieceType;
public PieceColor color;
public void ClickThis(){ }
}
}

namespace WaveCaveGames.UI{

public class Selection : MonoBehaviour {
public GameObject[] buttons;
public string key;

```



```

public void SetKeyValue(int value){}
public void BackupValue(){ }
public void SaveBackupValue(){ }
}

public class RandomSpriteLoader : MonoBehaviour {
public Sprite[] sprites;
}
}

namespace WaveCaveGames.Utilities{

public class ArrayUtility {
//Return the size of the array.
public static int Size<T>(T[] array){}
//Add an object to target array.
public static void IncreaseArray<T>(ref T[] array, T item){}
//Add an object to target array, and place it to target 'itemIndex'. Index must be equal or smaller than the size of the array.
public static void IncreaseArray<T>(ref T[] array, T item, int
    itemIndex){}
//Add multiple objects to target array.
public static void IncreaseArray<T>(ref T[] array, params T[]
    items){}
//Add a 'not null' object to target array.
public static void IncreaseArrayAvoidNull<T>(ref T[] array, T
    item){}
//Add a 'not null' object to target array, and place it to target 'itemIndex'.
public static void IncreaseArrayAvoidNull<T>(ref T[] array, T
    item, int itemIndex){}
//Remove object 'index' in target array. If the index is 2, it
    'll remove the second object. The index must be lower than the
    size of the array.
public static void DecreaseArray<T>(ref T[] array, int index){}
//Remove object from - to in target array. From must be lower
    than to. If the from is equal to to, it'll remove this object
    only.
public static void DecreaseArray<T>(ref T[] array, int from, int
    to){}
//Remove 'item' in target array. If you have multiple 'item',
    it'll remove the first one.
public static void DecreaseArrayObject<T>(ref T[] array, T item){}
}
}

```

```

//Remove from - to in target array. Make sure not have same it
ems in the array, or it'll not work correctly.
public static void DecreaseArrayObject<T>(ref T[] array, T fro
m, T to){}
//Remove all 'item' in target array. If the item is null, it's
recommend using RemoveEmptyItems.
public static void RemoveAll<T>(ref T[] array, T item){}
//Resize an array. This works same as the editor array resizin
g.
public static void Resize<T>(ref T[] array, int size){}
//Clear the array.
public static void ClearArray<T>(ref T[] array){}
//Remove all empty items in an array.
public static void RemoveEmptyItems<T>(ref T[] array){}
//Remove same items. Sometimes it'll not remove completely for
some strange null reference. To remove completely, call Remov
eEmptyItems.
public static void RemoveSameItems<T>(ref T[] array){}
//Return the index of the object 'item'. If you have multiple
'item', it'll return the first one. If not found, it'll return
-1.
public static int ObjectToIndex<T>(T[] array, T item){}
//Return all the array items' name string.
public static string ArrayItems<T>(T[] array){}
//Return the layouted array items' name string.
public static string ArrayItemsLayouted<T>(T[] array){}
}
}

```

# Tips

- All scripts are free to use in commercial games, but you can't sell it elsewhere, or publish this game without modification.
- You can easily use the provided method to create individual mode.
- If you don't like my checkers model, you can change another model. And don't forget the check size configuration.
- If the check trigger is unresponsive, drag the camera closer to checkers board,