



"Thanks to people who help fix my game"

My teacher **Andrew** who fixed my errors, and continue helping until this game is ready

Michael Cameron for helping and teaching me how to create a basic camera 2d for my game

Brian Crockford who teamed with me and Michael previous project and teach me how to design a basic parent child class

Developed by:

Dexter

100671953

TABLE OF CONTENTS

1. Introduction

2. Controls

3. Gameplay

4. Key Features

INTRODUCTION

This game is a side-scrolling game similar to Contra games, where players fight through waves of enemies and encounter enemies as they proceed in the level. Player fights through lots of enemies where enemies are trying to attack players. Players can dodge the enemies, boost, jump and shoot at the enemies.

This is the first version of the game, so there will be boss fight, players can get different kinds of weapons, and develop new skills, now you can change your weapon in keyboard by pressing page down and page up will change its weapons. There will be more weapons to use and upgrade its weapons and meet new characters.

CONTROLS

This game is supported both XBOX controller and PC keyboard and mouse. By default if player did not plug in their XBOX controller, it will automatically set it to keyboard and mouse, else if a XBOX controller is plug in, it will set to the new controller.

Keyboard And Mouse

W = move player upward
S = move player downward
A = move player left
D = move player right
Right Mouse = shoot bullet
SPACE = jump
SHIFT = boost speed

For debug purpose

Plus Sign = increase player health
Minus Sign = decrease player health
Page UP = change player weapon (normal)
Page DOWN = change player weapon (laser)

XBOX Controller

D-UP = move player upward
D-DOWN = move player downward
D-LEFT = move player left
D-RIGHT = move player right
X = shoot bullet
A = jump
RT = boost speed

GAMEPLAY



MAIN MENU

This is the main menu of the screen, where player can navigate where to play single player, multiplayer, change its option, or exit the game..

PLAY - Play in single campaign, where player fight through levels and waves of enemies alone.

MULTIPLAYER - Play in two or more players, where players can join and help each other fighting the enemies.

OPTIONS - change other options of the game

EXIT - exits the game.



SINGLE PLAYER SCREEN

This is the main game play of the screen, player above is its health, it always stays with you, wherever you go. enemies move toward to the player, and player can shoot the enemies. There are two weapons

MAIN WEAPON



more speed, default attack damage

OTHER WEAPONS



less speed, 2x default attack damage



MULTIPLAYER SCREEN

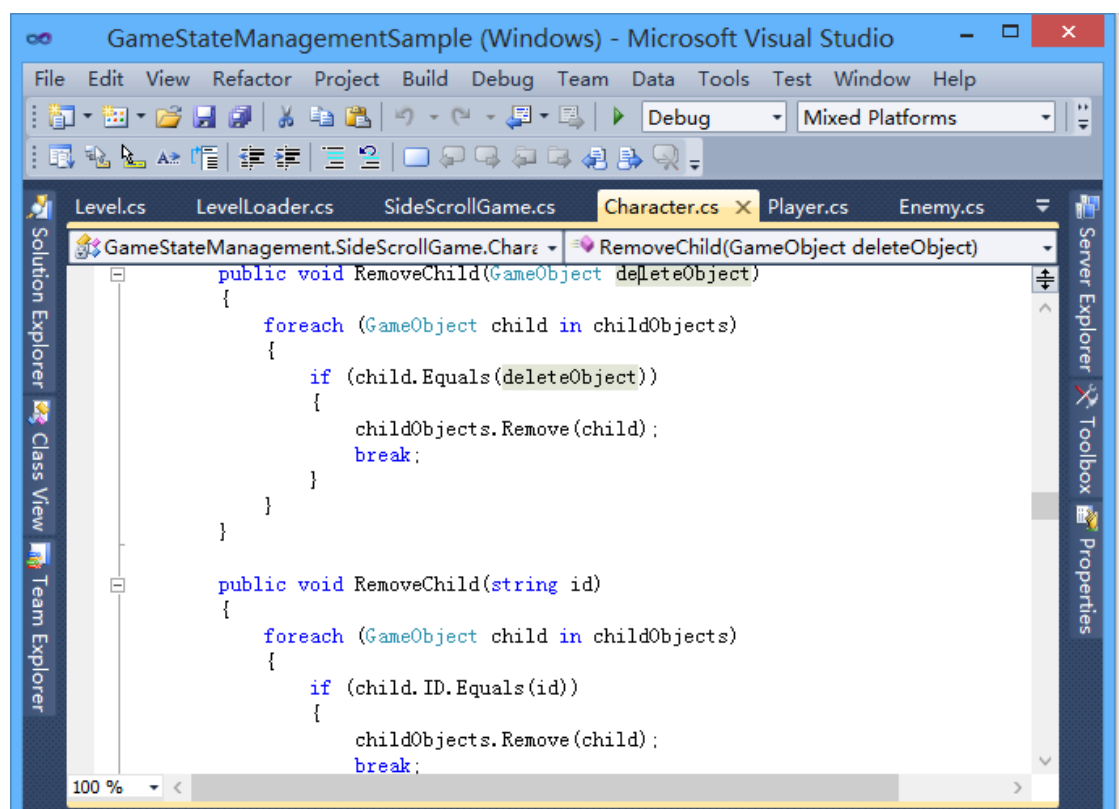
This is the multiplayer screen where players can help and fight through levels of enemies

KEY FEATURES

1. Organize Game objects

- this uses the parent and child class, which there is a parent class, and gameobjects such as weapon and health are child to the parent class, so you can add any little game object attach to the player or enemy that follows where player or enemy position. Thanks to Brian for teaching me how to organize this game objects. Child game objects can be add and remove by the player when player does not need it.

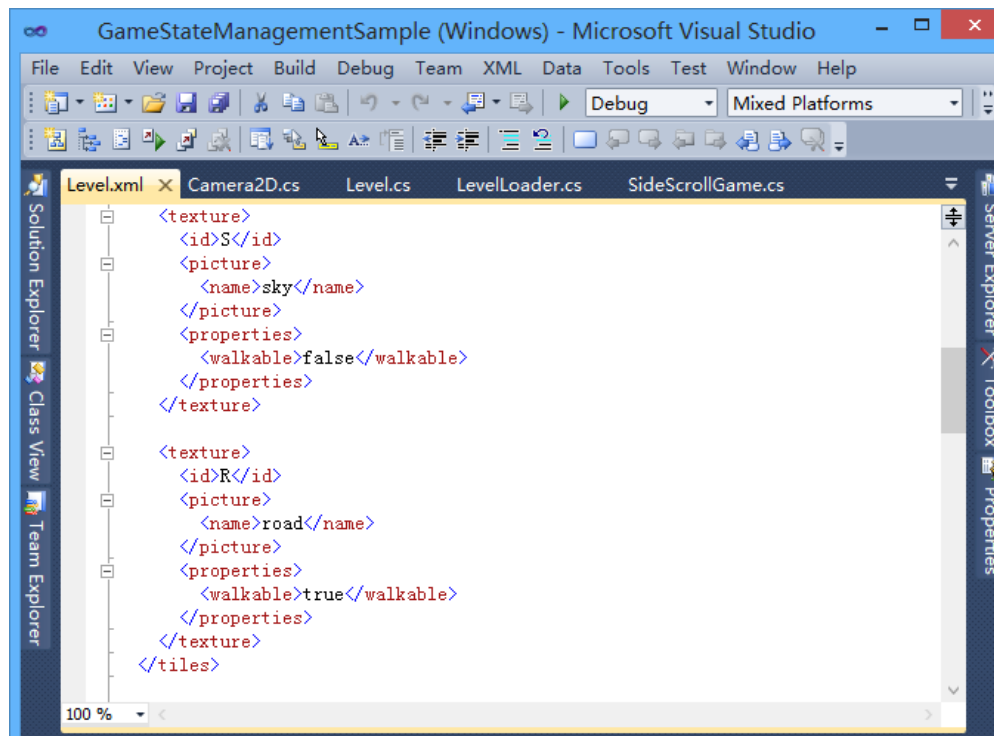
Every game object are the child class to character, you can find this on the Character.cs showing the parent and child class. Game object must have its own unique id that player can remove it by either searching the game object name or by searching its id. You can search any game object and edit its position by using the searching function of game object.



(shows in character.cs that you can remove class by searching id of its game object)

2. Level loader

- this level loader shows the loading its assets and putting it to the screen, you can change the level in the level.xml that shows here below.



(shows how to put a texture as a background for the side scroll game)

The picture above shows the requirement of adding its texture background for game to load. Every texture must have its own id, that you can call when to place it on the game (which will show it later in the manual), you can edit too its walkable to true or false, making the player can either walk that texture or not. Below this is the picture that shows in the level.xml of the enemy

```

<enemy>

  <kind>
    <id>N</id>
    <name>Normal</name>
  </kind>

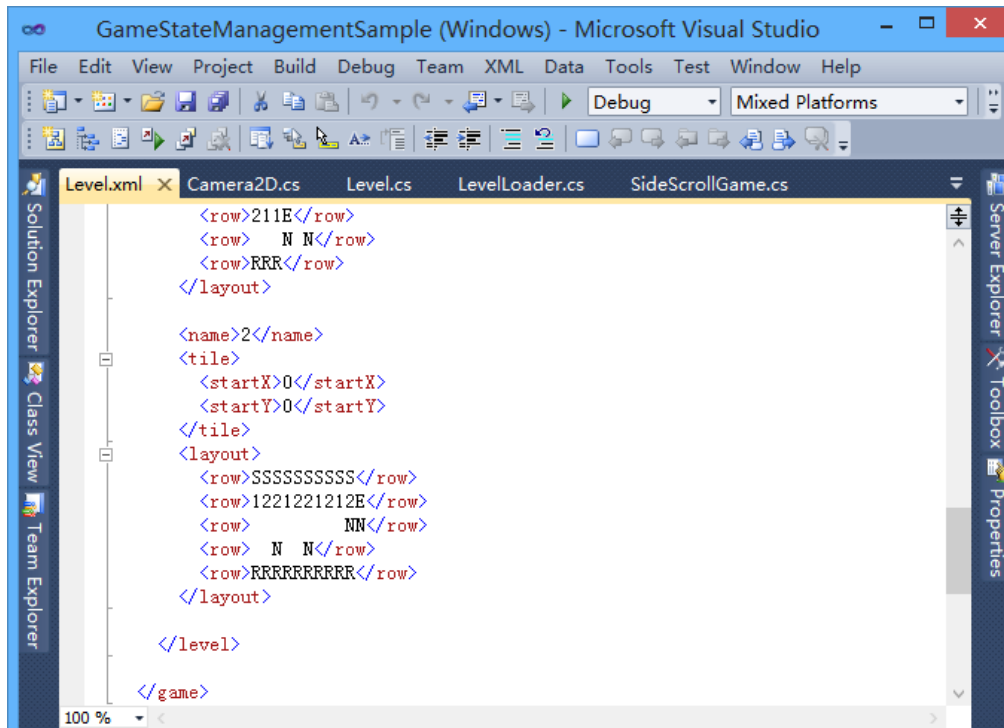
</enemy>

<level>

  <name>1</name>
  <tile>
    <startX>0</startX>
    <startY>0</startY>

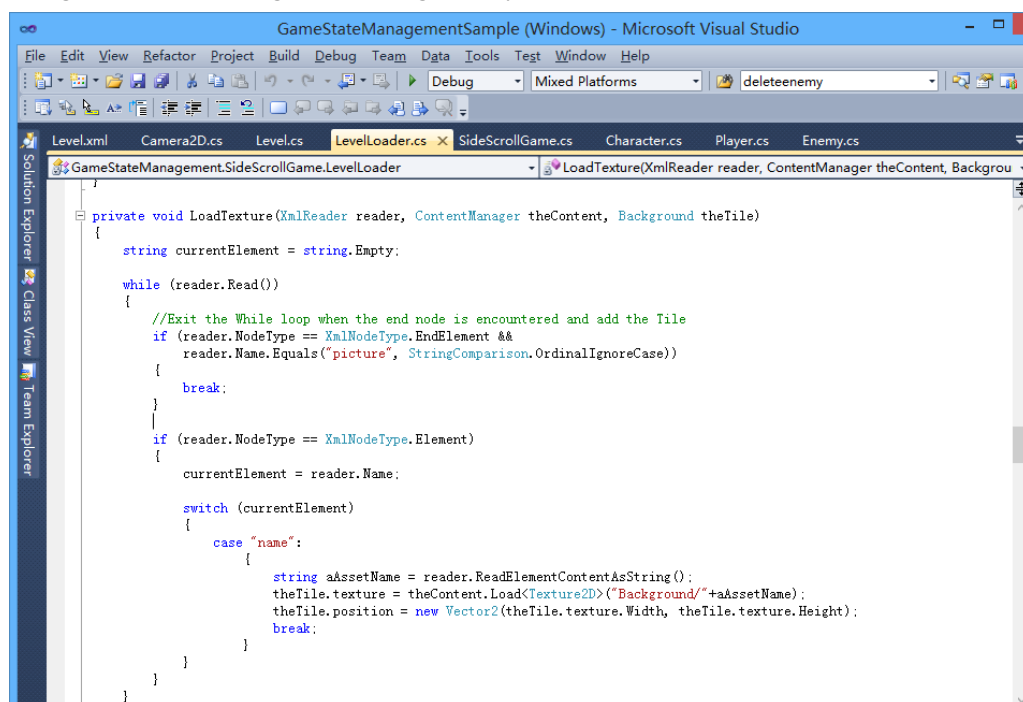
```

(shows the requirement of creating a new enemy asset that game can use it loading the level)



(shows on how to creating a new level that side scroll game can read and use draw its position by this xml)

the picture above shows the loading a level from level.xml that when player is in the certain level, level loader will load this level by its position of each id (like S "SKY", 1 "building1", N "Enemy Normal"), after setting the enemy position to the level, it must end by "E" to end this level width. As level loads, you can change its level colour, change the effects of the level, you can change it to black for night, blue for good sky, red for sunset and more..

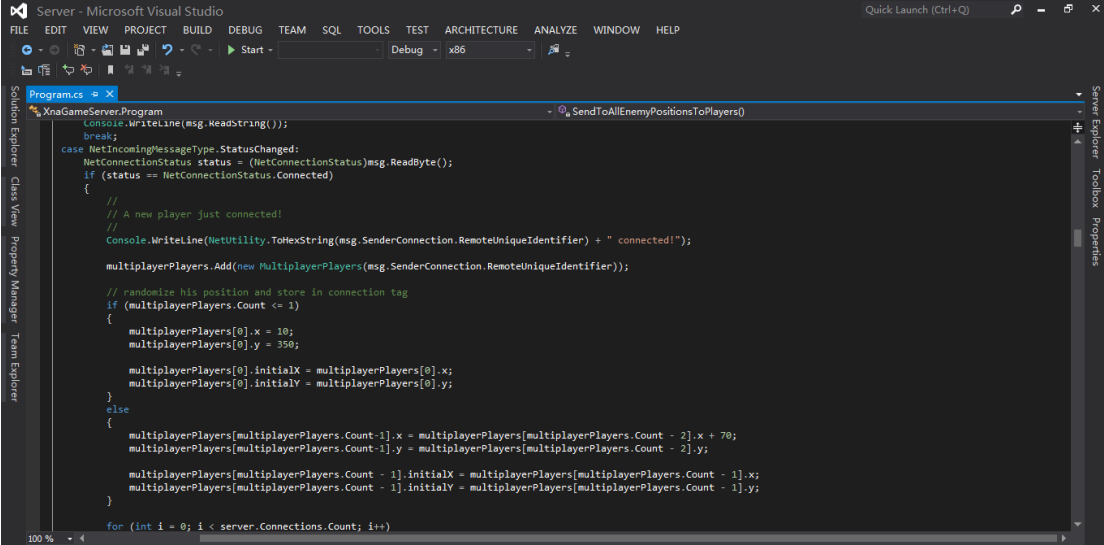


(shows on how level loader loads its texture) *LevelLoader.cs*

Networking Features

1. Any player can jump to the game

this game supports "NO" creating session, any player can just jump to the game, and help the other players, every client who joins or connect to the game, it delivers to all the players in the game, that a player was created and you should draw its player, and what player state is.



```
Server - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Start - Debug - x86
Program.cs
XnaGameServer.Program
SendToAllEnemyPositionsToPlayers()
Console.WriteLine(msg.ReadString());
break;
case NetIncomingMessageType.StatusChanged:
    NetConnectionStatus status = (NetConnectionStatus)msg.ReadByte();
    if (status == NetConnectionStatus.Connected)
    {
        // A new player just connected!
        Console.WriteLine(NetUtility.ToHexString(msg.SenderConnection.RemoteUniqueIdentifier) + " connected!");
        multiplayerPlayers.Add(new MultiplayerPlayers(msg.SenderConnection.RemoteUniqueIdentifier));
        // randomize his position and store in connection tag
        if (multiplayerPlayers.Count <= 1)
        {
            multiplayerPlayers[0].x = 10;
            multiplayerPlayers[0].y = 350;
            multiplayerPlayers[0].initialX = multiplayerPlayers[0].x;
            multiplayerPlayers[0].initialY = multiplayerPlayers[0].y;
        }
        else
        {
            multiplayerPlayers[multiplayerPlayers.Count - 1].x = multiplayerPlayers[multiplayerPlayers.Count - 2].x + 70;
            multiplayerPlayers[multiplayerPlayers.Count - 1].y = multiplayerPlayers[multiplayerPlayers.Count - 2].y;
            multiplayerPlayers[multiplayerPlayers.Count - 1].initialX = multiplayerPlayers[multiplayerPlayers.Count - 1].x;
            multiplayerPlayers[multiplayerPlayers.Count - 1].initialY = multiplayerPlayers[multiplayerPlayers.Count - 1].y;
        }
    }
    for (int i = 0; i < server.Connections.Count; i++)
```

(shows how the server creates a player when a player jumps to the game)

Any player can just jump it even when the player is in the middle of the level, it reads the last player position and add its texture and position width to your position.

2. Players can just leave the game without ending the session

Players can not only jump in to the game, but they can just leave the session without affecting players playing the game.

```
Server - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYSIS
Start Debug x86
Program.cs
XnaGameServer.Program
else if (status == NetConnectionStatus.Unconnected || status == NetConnectionStatus.Unconnecting)
{
    Console.WriteLine(NetUtility.ToHexString(msg.SenderConnection.RemoteUniqueIdentifier) + " DISCONNECTED FROM SERVER!");

    for (int i = 0; i < multiplayerPlayers.Count; i++)
    {
        if (multiplayerPlayers[i].id == msg.SenderConnection.RemoteUniqueIdentifier)
        {
            if (multiplayerPlayers[i].isHost)
            {
                if (multiplayerPlayers.Count > 1)
                {
                    multiplayerPlayers[i + 1].isHost = true;

                    NetConnection player = server.Connections[i];

                    NetOutgoingMessage outMsg = server.CreateMessage();
                    outMsg.Write((byte)PacketTypes.CHANGEHOST);
                    outMsg.Write((bool)multiplayerPlayers[i + 1].isHost);

                    server.SendMessage(outMsg, player, NetDeliveryMethod.ReliableOrdered);
                }
            }

            multiplayerPlayers.RemoveAt(i);
            if (deletePlayerFromServer(msg.SenderConnection.RemoteUniqueIdentifier))
            {
                Console.WriteLine(NetUtility.ToHexString(msg.SenderConnection.RemoteUniqueIdentifier) + " DELETED!");
            }
            else
            {
                Console.WriteLine(NetUtility.ToHexString(msg.SenderConnection.RemoteUniqueIdentifier) + " IS NOT EXIST!");
            }
        }

        foreach (NetConnection player in server.Connections)
        {
            NetOutgoingMessage outMessage = server.CreateMessage();
            outMessage.Write((byte)PacketTypes.DELETEPLAYER);
            outMessage.Write((long)msg.SenderConnection.RemoteUniqueIdentifier);

            server.SendMessage(outMessage, player, NetDeliveryMethod.ReliableOrdered);
        }

        SetEnemyTarget();
        SendToAllPlayerEnemyTarget();

        break;
    }
}
```

(shows when player leaves the game)

if the player is the host of the game and leaves, the next player become the host for the players, and updates the server of enemy position.