Platformer Directions – Modified due to new Unity version, there may be some conversion issues so let me know if you have any problems.

1. File-New project – Platformer\_Demo or whatever you want, select 2d
2. Give it time, left click on the center map/scene
3. GameObject- new empty
4. Add component, sprite renderer, black box, make it a larger size using the scale tool to see
5. Add component box collider 2d
6. Add component rigidbody2d, set linear drag to 2, angular drag to 2, constraints > freeze rotation z
7. New gameobject under that one, scale to a larger size and name it “CenterPlatform”, addcomponent sprite renderer, white box, box collider 2d
8. Play using play button in top
9. Right click on folder in corner, make new physics 2d material, name bounce, try out your own values and play around, but afterwards set bounciness to 0 (< 1 slows down, > 1 gains speed), friction to 0
10. Apply to ball
11. Make sure all objects have z = 0
12. Play
13. Click on smaller item, add component new script named BallController, Open script, in update type the following (case sensitive)
14. **void Update () {**
15. **if (Input.GetKey(KeyCode.A)) {**
16. **this.GetComponent<Rigidbody2D> ().AddForce(new Vector2(-30, 0));**
17. **} else if (Input.GetKey(KeyCode.D)) {**
18. **this.GetComponent<Rigidbody2D> ().AddForce(new Vector2(30, 0));**
19. **} else if (Input.GetKey(KeyCode.S)) {**
20. **this.GetComponent<Rigidbody2D> ().AddForce(new Vector2(0, -15))**
21. **}**
22. Rigidbody2D is what controls the intertia, velocity, mass, and other physical properties of objects.
23. Save, run
24. Open same script, add line for jumping
25. **if (Input.GetKeyDown(KeyCode.W)) {**
26. **this.GetComponent<Rigidbody2D> ().AddForce(new Vector2(0, 300));**
27. **}**
28. Notice the player has infinite jumps. Now you should add a counter as a global, private value named “jumpcounter”
29. **private int jumpcounter;**
30. **void OnCollisionEnter2D(Collision2D col) {**
31. **if (!col.gameObject.GetComponent</\*script name here\*/>()) jumpcounter = 0;**
32. **}**
33. **void OnCollisionExit2D(Collision2D col) { // to prevent two jumps if you fall off**
34. **if (!col.gameObject.GetComponent<BallController>()) jumpcounter++;**
35. **}**
36. **//change the jumping code**
37. **if (Input.GetKeyDown(KeyCode.W) && jumpcounter < 2) {**
38. **if (jumpcounter > 0) jumpcounter++;**
39. **this.GetComponent<Rigidbody2D>().velocity = new Vector2(this.** **GetComponent<Rigidbody2D>().x, 0);**
40. **this.GetComponent<Rigidbody2D>().AddForce(new Vector2(0, 300));**
41. **}**
42. In Start() type:
43. **void Start() {**
44. **This.tag = “Player”;**
45. **}**
46. Add methods for other players, create public value ID to keep track which player is which so we can have multiplayer
47. **int playerid;**
48. **bool GetLeftKey(float id) {**
49. **if (id == 0) return Input.GetKey(KeyCode.A);**
50. **if (id == 1) return Input.GetKey(KeyCode.J);**
51. **return false;**
52. **}**
53. **bool GetRightKey(float id) {**
54. **if (id == 0) return Input.GetKey(KeyCode.D);**
55. **if (id == 1) return Input.GetKey(KeyCode.L);**
56. **return false;**
57. **}**
58. **bool GetJumpKey(float id) {**
59. **if (id == 0) return Input.GetKeyDown(KeyCode.W); //keydown for the rest**
60. **if (id == 1) return Input.GetKeyDown(KeyCode.I);**
61. **return false;**
62. **}**
    1. **bool GetDownKey(float id) {**
63. **if (id == 0) return Input.GetKey(KeyCode.S);**
64. **if (id == 1) return Input.GetKey(KeyCode.K);**
65. **return false;**
66. **}**
67. **bool GetAttackKey(float id) {**
68. **if (id == 0) return Input.GetKeyDown(KeyCode.E);**
69. **if (id == 1) return Input.GetKeyDown(KeyCode.O);**
70. **return false;**
71. **}**
72. Replace input.getkey with the get keys, for example replacing **Input.GetKey (KeyCode.A)** with **GetLeftKey(playerid)**
73. Add another entity, make their color red, set player id to 1
74. Play around, watch them jump around each other
75. Add attack and takedamage functions
76. **public float damage;**
77. **if (GetAttackKey(playerid)) {**
78. **This.SendMessage(“Attack”);**
79. **}**
80. **// create child gameobject by making new gameobject and dragging it into one of the players, add circlecollider, set istrigger to true, do the same for both.**
81. **Add to ballcontroller script:**
82. **public void TakeDamage(float f, Vector2 placement) {**
83. **this.damage += f;**
84. **this.GetComponent<Rigidbody2D> ().AddForce(new Vector2(0, damage));**
85. **this.GetComponent<Rigidbody2D> ().AddForce (f \* damage \* ((Vector2)this.transform.position - placement));**
86. **}**
87. **//create new script within the child, called AttackZone**
88. **private bool inzone;**
89. **private GameObject focus;**
91. **void OnTriggerEnter2D(Collider2D col) {**
92. **if (col.GetComponent<BallController>() != null) {**
93. **inzone = true;**
94. **}**
95. **}**
96. **void OnTriggerStay2D(Collider2D col) {**
97. **if (col.GetComponent<BallController>() != null) focus = col.gameObject;**
98. **}**
99. **void OnTriggerExit2D(Collider2D col) {**
100. **if (col.GetComponent<BallController>() != null) {**
101. **inzone = false;**
102. **}**
103. **}**
104. **void Attack() {**
105. **if (inzone) {**
106. **focus.GetComponent<BallController>().TakeDamage(5, this.transform.position);**
107. **}**
108. **}**
109. Calling upon “sendMessage” lets you execute any public function in antother script, relative to the game object. You can also pass variables as another parameter, but as of now you cannot return variables.
110. Add attack cooldown time
     * Add two more public variables, one being “attackcooldowntime” and the other “attackcooldownsofar”
     * Change attack method to:
111. **this.coolDownAttack+= Time.deltaTime;**
112. **if (GetAttackKey(playerid) && coolDownAttack > this.damageCooldownTime) {**
113. **coolDownAttack = 0**
114. **this.transform.FindChild("TriggerZone").SendMessage("Attack");**
115. **}**
116. “TriggerZone” is what I named the child inside of the player
117. Add four boxes around the outer rims, each istrigger=true boxes, and add a script called “out of bounds”
118. **void OnTriggerEnter2D(Collider2D col) {**
119. **if (col.GetComponent<BallController>()) {**
120. **col.transform.position = respawnPoint;**
121. **col.GetComponent<Rigidbody2D> ().velocity = Vector2.zero;**
122. **col.GetComponent<BallController>().deaths++;**
123. **col.GetComponent<BallController>().damage = 0;**
124. **}**
125. **}**
126. Add new component, add guitext component, add any text and set font to 10, set to scene view and start changing values until you can find the words and set it to your desired place
127. Create guitext script and add
128. **void Update () {**
129. **string text = "";**
130. **GameObject[] Players = GameObject.FindGameObjectsWithTag("Player");**
131. **foreach (GameObject p in Players) {**
132. **text += "Player " + p.GetComponent<BallController>().playerid +**
133. **" " + p.GetComponent<BallController>().damage +**
134. **"% Deaths = " +p.GetComponent<BallController>().deaths + '\n';**
135. **}**
136. **this.** **GetComponent<GUIText>().text = text;**
137. **}**
138. Add component to camera, label it “CameraZoomer”
139. **public float dampTime = 0.15f;**
140. **private Vector3 velocity = Vector3.zero;**
141. **GameObject[] targets;**
142. **public float bufferX = 0, bufferY = 0; // you can change these to make the camera accurate, if needed**
143. **void Update ()**
144. **{**
145. **targets = GameObject.FindGameObjectsWithTag("Player");**
146. **if (targets != null)**
147. **{**
148. **float avgX = 0;**
149. **float avgY = 0;**
150. **int i = 0;**
151. **foreach (GameObject g in targets) {**
152. **avgX += g.transform.position.x;**
153. **avgY += g.transform.position.y;**
154. **i++;**
155. **}**
156. **avgX += GameObject.Find("CenterPlatform").transform.position.x;**
157. **avgY += GameObject.Find("CenterPlatform").transform.position.y;**
158. **i++;**
159. **Vector3 avg = new Vector3(avgX/i, avgY/i, this.transform.position.z);**
160. **print(avg);**
161. **dampTime = 1/Vector2.Distance(this.transform.position, avg);**
162. **Vector3 point = GetComponent<Camera>().WorldToViewportPoint(avg);**
163. **Vector3 delta = avg - GetComponent<Camera>().ViewportToWorldPoint(new Vector3(.05f, .05f, point.z));**
164. **Vector3 destination = transform.position + delta;**
165. **destination.Set (destination.x + bufferX, destination.y + bufferY, destination.z);**
166. **transform.position = Vector3.SmoothDamp(transform.position, destination, ref velocity, dampTime);**
168. **}**
169. **}**
170. If you don’t understand what this code implies, it makes the camera follow every player in the arena. If you need more help to understand it, feel free to ask me.
171. There we go, your new game is fun to play. Feel free to edit it around, and enjoy yourself to add your own little things.
172. Known bugs:
     * Can double jump again if you hit the edge of the platform (can you fix this?)