－－－－－－－－－－－－－－－－－－－

1.首先我们要在GameScene.sks对场景进行编辑，可以在右边的第三栏看到场景size的设置，将size大小调整到合理范围

－－－－－－－－－－－－－－－－－－－

2.之后我们需要设置背景，从Object Library中拖取Color Sprite 到视窗中，并调整位置（position），将texture设置为bg（背景图）

－－－－－－－－－－－－－－－－－－－

3.在 GameScene.sks，拖一個 Color Sprite 到場景中。將它命名為 ball，Texture 設置為 ball，Position 設置為 284，220，Z Position 設為 2，確保小球顯示在背景的上層。（基本操作和放置背景一样）

－－－－－－－－－－－－－－－－－－－

4.這裏，你創建了一個立方體類型的物理體，是一個圓，大小和 ball 貼圖相同。這個物理體受外力和衝量影響，能夠和其它物理體發生碰撞。

它有這些屬性：

1. Allow Rotation 的作用是允許物體旋轉或不旋轉。這裏，我们不需要球體旋轉。
2. Friction 的作用是设置有无摩擦，這裏簡單地設置為沒有摩擦。
3. Restitution 它就是對象的彈性。設置為 1，表示球體和其它物體碰撞後，將表現為絕對彈性。簡單説，球體會以相同的力彈回去。
4. Linear Damping 模擬了液體或空氣對物體線性速度的遞減程度。在 Breakout 遊戲中，球體在移動過程中速度不會衰減，因此設置為 0。
5. Angular Damping 和 Linear Damping 一樣，但針對的是角速度。這個設置不是必須的，因為你根本不允許旋轉。

注意：通常，最好讓物理體和玩家看到的非常接近。對於球體，這是很容易模擬的。但是，越複雜的形狀你就越要花心思，因為複雜物體確實比較耗性能。從 iOS 8 和 Xcode 6 開始，Sprite Kit 支持

alpha 遮罩物體類型，自動把 Sprite 的形狀作為物理體的形狀，但需要注意的是，這會降低性能。

－－－－－－－－－－－－－－－－－－－

5.

1. 創建了一個邊界物體。和球體的立方體類型的物體不同，邊界物體沒有體積和質量，不受外力或衝量的影響。
2. Friction 設置為 0，這樣球體與之碰撞後不會速度衰減。相反，你需要的是絕對反彈，這樣會以擊中時相同的角度離開。
3. 你可以為每個節點設置物理體。這裏，你將邊界物體賦給場景。注意：SKPhysicsBody 的座標是相對於節點位置的。

－－－－－－－－－－－－－－－－－－－

6.運行項目，你會看到球體和之前一樣下落，但現在會在“籠子”的下端彈回。因為我們將籠子和環境中的摩擦力去掉了，同時將恢復設置為完美彈性體，小球會無限地反彈下去。

－－－－－－－－－－－－－－－－－－－

7.從場景中去掉重力加速度，然後從場景的子節點中通過名字（這個名字是你可以通過可視化編輯器進行設置）來獲取小球，然後施加一個力。這個力將使物理體沿給定方向運動（在這裏，即右下 45 度方向）。當小球開始運動後，因為你添加的“籠子”它只能在屏幕上來回反彈了。

－－－－－－－－－－－－－－－－－－－

8.這些屬性大部分和小球類似。只不過這次的物理體是 Bounding rectangle，因為它和長方形的板子更匹配。

這裏反選了 Dynamic，這樣木板就變成靜態物體了。這使得木板不會受力和衝量的影響。待會你就會明白這一點很關鍵。

－－－－－－－－－－－－－－－－－－－

這個屬性用於保存玩家是否觸摸到了木板上。你在實現模板的拖拽時需要用到它。

－－－－－－－－－－－－－－－－－－－

在 GameScene 中通過内置的touch 回調方法來實現檢測觸摸

－－－－－－－－－－－－－－－－－－－

9.這個方法獲取了觸摸，並找到它在屏幕上的位置。然後用 body(at:) 方法查找該位置是否存在一個物理體。然後，判斷觸摸位置是否有節點，如果有，這個節點是不是就是我們要找的木板。早先我們已經設置過這個對象的名字——你可以通過 name 屬性判斷對象是否是我們想要找的對象。如果觸摸位置是木板，輸出文本消息，然後將 isFingerOnPaddle 方法設置為 true。

－－－－－－－－－－－－－－－－－－－

10.

1. 判斷玩家是否已經觸摸到了木板。
2. 如果是，根據玩家手指的移動更新木板的位置。這需要獲取當前觸摸位置和上一次的觸摸位置。
3. 獲取木板對應的 SKSpriteNode。
4. 用木板當前位置加上兩次觸摸位置之差。
5. 在移動木板之前，限制它的位置，防止它移出屏幕左右兩邊。
6. 根據之前計算的結果，設置木板的位置。

－－－－－－－－－－－－－－－－－－－

11.這裏的 isFingerOnPaddle 設為 false。這確保了玩家從屏幕上拿走他們的手指，然後又再次點擊木板時，木板不會跳到之前觸摸的位置。

－－－－－－－－－－－－－－－－－－－

12.這段代碼簡單地為之前創建的物理體設置了 categoryBitMask。

－－－－－－－－－－－－－－－－－－－

13.categoryBitMask: 這個位掩碼用於識別物體屬於哪一類別。你可以用它定義物體能夠和其它物體進行交互。

contactTestBitMask: 設置這個位掩碼會讓物體和其它物體（指定某個 category 的）發生碰撞時通知委託對象。

修改的目的是让GameScene可以接收所有指定的物體的碰撞通知。

这样碰撞部分就可以了

－－－－－－－－－－－－－－－－－－－

14.這段代碼創建了 8 塊磚，並放在屏幕中央。

1. 这一段代码设置了磚塊的數目以及它們的寬（即砖块的一些常量）。
2. 計算 x 偏移。這是第一塊磚和屏幕左邊沿的距離。用屏幕寬度減去8塊磚的總寬度再除以 2。
3. 創建磚塊，設置每塊磚的物理屬性，並根據 blockWidth 和 xOffset 設置每塊磚的位置。

－－－－－－－－－－－－－－－－－－－

15.通過定義這個變量，你為遊戲創建了一個狀態機。（状态机的作用是管理一连串的状态动作，并且执行成循环回圈）

－－－－－－－－－－－－－－－－－－－

16.WaitingForTap 狀態是遊戲剛加載等待開始的狀態。玩家會看到一個 Tap to Play 的提示，遊戲等待觸摸事件一發生就會進入 play 狀態。

這裏創建了一個 sprite 用來顯示 Tap to Play 文本，後面則會用來顯示 Game Over。然後告訴狀態機進入 WaitingForTap 狀態。

同時在 didMove(to:) 方法中刪除這句：

－－－－－－－－－－－－－－－－－－－

17.Playing 狀態會開始遊戲，並管理小球的速度。

－－－－－－－－－－－－－－－－－－－

18.助手方法可以返回一個位於兩個參數之間的隨機數。你會用它來讓小球一開始的方向產生一些隨機性。（注：25是可以让添加的助手进行动作，26是添加助手这一行为）

－－－－－－－－－－－－－－－－－－－

19.此处在遊戲進入 Playing 狀態时，獲取小球 sprite，調用它的 applyImpulse(\_:) 方法，讓它開始移動。

－－－－－－－－－－－－－－－－－－－

20.update(deltaTime:) 方法會在每一幀的 Playing 狀態時調用。獲得小球對象，判斷它的速度，如果小球在某个地方卡住，或速度为低于一定的值，则会额外给小球一个力让它重新回到一定角度的运动。

－－－－－－－－－－－－－－－－－－－

21.當遊戲進入 GameOver 狀態，設置了小球的線性阻力和重力加速度，調至小球掉到地板上並逐漸變慢。

這就是遊戲結束狀態。

－－－－－－－－－－－－－－－－－－－

22.這個方法通過遍歷場景中的所有子節點檢查場景中還剩下幾塊磚，

如果一塊磚都沒有了，判定玩家勝，返回 true。

－－－－－－－－－－－－－－－－－－－

23.當球碰到屏幕底部，遊戲結束。

－－－－－－－－－－－－－－－－－－－