CHAPTER 28

PROTOTYPE 1: APPLE PICKER FOR UNITY 5.X

In the pages below, I've replaced the sections of Chapter 28 that used GUIText with new pages that take advantage of the UGUI (Unity Graphical User Interface) system that first appeared in Unity 4.6.

These pages are not the whole chapter and only replace parts of pages 440-442 and 445-446.

– Jeremy Bond

(Several pages of the chapter are skipped here.)

GUI and Game Management

The final things to add to our game are the GUI and *game management* that will make it feel like more of a real game. The GUI element we'll add is a score counter, and the game management elements we'll add are levels and lives.

Score Counter

The score counter will help players get a sense of their level of achievement in the game.

Open _Scene_0 by double-clicking it in the Project pane. Then go to the menu bar and choose *GameObject > UI > Text*. Because this is the first UGUI (Unity Graphical User Interface) element to be added to this scene, it will add several things to the Hierarchy pane. The first you'll see is a *Canvas*. The Canvas is the 2-dimensional board on which the GUI will be arranged. Looking in the Scene pane, you should also see a **very** large 2D box extending from the origin out very far in the x and y directions. Double click on Canvas in the Hierarchy to zoom out and see the whole thing. This will be scaled to match your Game pane, so if you have the Game pane set to a 16:9 aspect ratio, the Canvas will follow suit.

The other GameObject added at the top level of the Hierarchy is the *EventSystem*. The EventSystem is what allows buttons, sliders, and such that you build in UGUI to work, however, we will not be making use of it in this prototype.

Below the Canvas, you will see a *Text* GameObject. Double-click on the Text GameObject in the Hierarchy pane to zoom in on it. It is very likely that the text color defaulted to black, which may be difficult to see over the background of the Scene pane. Select the Text GameObject in the Hierarchy and use the Inspector pane to change its name to *HighScore*. Follow these directions to make the HighScore Inspector match that shown in Figure 28.10:

1. In the *RectTransform* component of the Inspector:
   - Set Anchors Min X=0, Min Y=1, Max X=0, and Max Y=1.
   - Set Pivot X=0 and Y=1.
   - Set Pos X=10, Pos Y = -6, and Pos Z = 0.
   - Set Width=256 and Height=32.
   - After doing this, you may want to double-click HighScore in the Hierarchy again to re-center it in the Scene pane.
2. In the *Text (Script)* component of the Inspector:
   - Set the text to "High Score: 1000" (without the quotes around it).
   - Set the Font Style to Bold.
   - Set the Font Size to 28.
   - Set the Color to white, which will make it much more visible in the Game pane.

Now, right-click on HighScore in the Hierarchy and choose Duplicate. Select the new HighScore (1) GameObject and change its name to ScoreCounter. Then alter its values in the Inspector to match those shown in Figure 28.10. Don't forget to set the Anchors and Pivot in the RectTransform component and the Alignment in the Text component. You'll notice that when you change the Anchors or Pivot in the RectTransform, Unity automatically changes the Pos X to keep the ScoreCounter in the same place within the Canvas. To prevent Unity from doing this, click the R button in the RectTransform that is shown selected in the ScoreCounter Inspector in Figure 28.10.

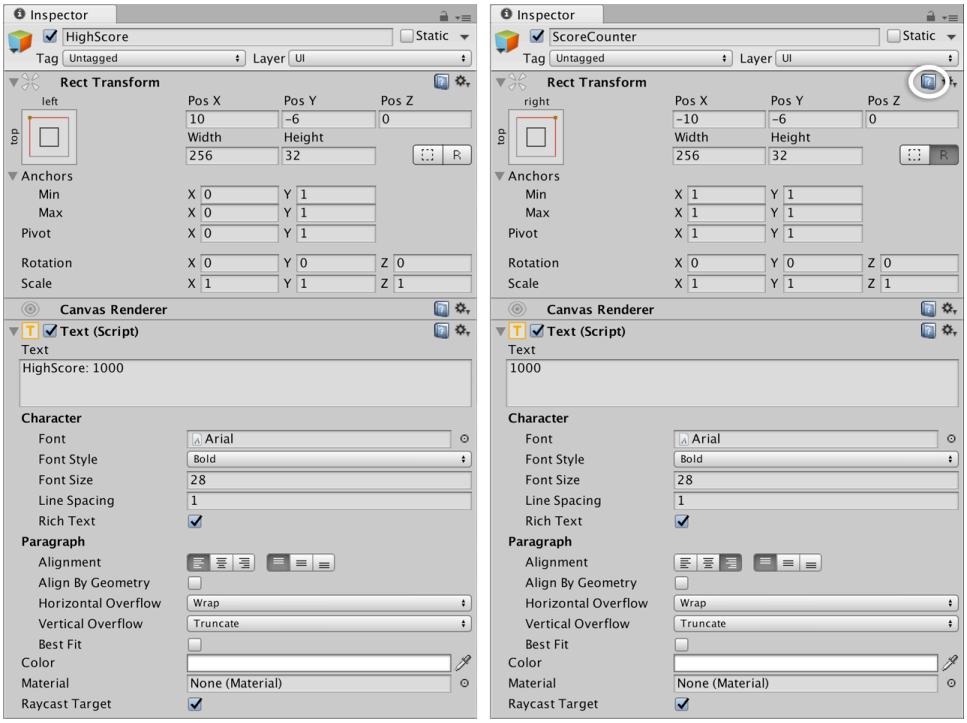

Figure 28.10 RectTransform and Text component settings for HighScore and ScoreCounter

As you've seen here, the coordinates for UGUI GameObjects differ completely from those for regular GameObjects and use a RectTransform instead of a regular Transform. The coordinates for a RectTransform are all relative to the Canvas parent of the UGUI GameObject. Clicking the help icon for the RectTransform component (circled in Figure 28.10) can give you more information about how this works.

Add Points for Each Caught Apple

There are two scripts that are notified when a collision occurs between an apple and a basket: the Apple and Basket scripts. In this game, there is already an `OnCollisionEnter()` method on the Basket C# script, so we'll modify this to give the player points for each apple that is caught. 100 points per apple seems like a reasonable number (though I've personally always thought it was a little ridiculous to have those extra zeroes at the end of scores). Open the Basket script in MonoDevelop and add the bolded lines shown here:

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;     // This line enables use of UGUI features.        // 1


public class Basket : MonoBehaviour {

    public Text           scoreGT;                                           // 1

    void Update () {
        …
    }

    void Start() {
        // Find a reference to the ScoreCounter GameObject
        GameObject scoreGO = GameObject.Find("ScoreCounter");                // 2
        // Get the Text Component of that GameObject
        scoreGT = scoreGO.GetComponent<Text>();                              // 3
        // Set the starting number of points to 0
        scoreGT.text = "0";
    }
```

```
    void OnCollisionEnter( Collision coll ) {
        // Find out what hit this Basket
        GameObject collidedWith = coll.gameObject;
        if ( collidedWith.tag == "Apple" ) {
            Destroy( collidedWith );
        }

        // Parse the text of the scoreGT into an int
        int score = int.Parse( scoreGT.text );                              // 4
        // Add points for catching the apple
        score += 100;
        // Convert the score back to a string and display it
        scoreGT.text = score.ToString();
    }
}
```

1. Be sure you don't neglect to enter these lines. They are separated from the others.
2. `GameObject.Find("ScoreCounter")` searches through all the GameObjects in the scene for one named "`ScoreCounter`" and assigns it to the local variable `scoreGO`.
3. `scoreGO.Getcomponent<Text>()` searches for a Text component on the scoreGO GameObject, and this is assigned to the public field `scoreGT`. The starting score is then set to zero on the next line. Without the `using UnityEngine.UI;` line above, the Text component would not be defined for C# within Unity. As Unity Technology's coding practices get stronger, they are moving to more of a model like this where you must include the code libraries for their new features manually.
4. `int.Parse( scoreGT.text )` takes the text shown in ScoreCounter and converts it to an integer. 100 points are added to the int score, and it is then assigned back to the text of scoreGT after being parsed from an int to a string by `score.ToString()`.

(Several pages of the chapter are skipped here.)

Adding a High Score

Now we'll make use of the HighScore Text that you created earlier. Create a new C# script named *HighScore*, attach it to the HighScore GameObject in the Hierarchy pane, and give it the following code:

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;    // Remember, we need this line for UGUI to work.

public class HighScore : MonoBehaviour {
    static public int    score = 1000;

    void Update () {
        Text gt = this.GetComponent<Text>();
        gt.text = "High Score: "+score;
    }
}
```

The lines in `Update()` simply display the value of `score` in the Text component. It is not necessary to call `ToString()` on the `score` in this instance because when the `+` operator is used to concatenate a string with another data type (the "`High Score: `" string literal is concatenated with the int `score` in this case), `ToString()` is called implicitly (that is, automatically).

Making the int `score` not only public but also static gives us the ability to access it from any other script by simply typing `HighScore.score`. This is one of the powers of static variables that we will use throughout the prototypes in this book. Open the Basket C# script and add the following lines to see how this is used:

```
    void OnCollisionEnter( Collision coll ) {
        …
        // Convert the score back to a string and display it
        scoreGT.text = score.ToString();

        // Track the high score
        if (score > HighScore.score) {
            HighScore.score = score;
        }
    }
```

Now `HighScore.score` is set any time the current score exceeds it.