Code Revision

Jason Ege

Piece of code that shows I updated something in the game to make it work better

This is example code from the car game I made with Tyler. It shows simplification of code that made modifying the behavior of the car easier as well as making the car behavior more realistic in-game.

- Originally I controlled the car by applying forces. That worked okay for starts…

```
void ControllerByForce()
	{
		if (Input.GetKey (KeyCode.W) && transform.constantForce.force.z < 8.0f) {
			rigidbody.AddForce(transform.forward *100,ForceMode.Force);
			//		transform.constantForce.force += new Vector3 (transform.forward.x, transform.forward.y,
transform.forward.z);
		}
		else if (Input.GetKey (KeyCode.S) && constantForce.force.z > -6.0f) {
			rigidbody.AddForce(-transform.forward *100,ForceMode.Force);
			//transform.constantForce.force -= new Vector3 (transform.forward.x, transform.forward.y,
transform.forward.z);
		}
		else if (!Input.GetKey (KeyCode.W) && !Input.GetKey (KeyCode.S)) {
			if (transform.constantForce.force.z > 0.0f)
			{
				transform.constantForce.force -= new Vector3(transform.forward.x, 0.0f,
transform.forward.z);
			}
			else if (transform.constantForce.force.z < 0.0f)
			{
				transform.constantForce.force += new Vector3(transform.forward.x, 0.0f,
transform.forward.z);
			}
		}
	}
```

- …But then I tried using variables for acceleration, deceleration, maximum speed allowed, etc. This gave me more control over the car's behavior, and simplified the code. An example of how it simplified the code is that the application of the variables is now condensed into one line, making it easier to understand and modify. I also used public variables so that one can edit the variables within the unity interface rather than having to look at the code.

```
public float acceleration;
public float deceleration;
public float MaxSpeed;
float speed;

void ControllerByTranslate()
	{
		if (Input.GetKey (KeyCode.W))
		{
			if (speed < MaxSpeed)
			{
				speed += acceleration;
			}
		}
		if (Input.GetKey (KeyCode.S))
		{
```

```
                        if (speed > -0.025)
                        {
                                speed -= acceleration;
                        }
                }
                if (!Input.GetKey (KeyCode.W) && !Input.GetKey (KeyCode.S))
                {
                        if (speed > 0.005f)
                        {
                                speed -= deceleration/2;
                        }
                        else if (speed < -0.005f)
                        {
                                speed += deceleration/2;
                        }
                        else if (speed >= -0.005f && speed <= 0.005f)
                        {
                                speed = 0.000001f;
                        }
                }
                transform.Translate (new Vector3(transform.forward.x*speed, 0.0f,
transform.forward.z*speed),Space.World);
        }
```

Code Revision

Deven Smith

An example of where I had to revise my code comes from the first prototype that I made, Time Shooter. In it I can shoot objects to make them age. One of those objects is a tree. Originally I was just scaling an objects size based on whether it was younger or older. An example of that is shown below.

```
void Aged (int effect)
	{
		if(effect > 0)
		{
			if(age == Ages.superOld)
				Destroy (this.gameObject);
			else
				age++;
		}
		else if (effect < 0)
		{
			if(age == Ages.baby)
				Destroy (this.gameObject);
			else
				age--;
		}

		switch(age)
		{
		case Ages.baby:
			transform.localScale = new Vector3(.25f,.5f,.25f);
			break;
		case Ages.young:
			transform.localScale = new Vector3(.25f,3,.25f);
			break;
		case Ages.normal:
			transform.localScale = new Vector3(.75f, 7f, .75f);
			break;
		case Ages.old:
			transform.localScale = new Vector3(2f,12f,2f);
			break;
		case Ages.superOld:
			transform.localScale = new Vector3(3f,24f,3f);
			break;
		default:
			transform.localScale = new Vector3(.5f,1,.5f);
			break;
		}
	}
```

In that code I had preset scales that an object would become based on what age it had became. I realized that this was the wrong approach about the time I received the art assets. A tree at its youngest is not a really small adult tree it is an acorn in this game. To make this work for the revised code I separated the aging process and how the tree appears at an age into two functions. Since the trees went from being a single object into a series of planes with an image on them I had to switch all the art on each plane that forms the tree. After that I needed to change the collider size and center for each age of tree and determine if the tree needed to be active or the acorn. The revised code appears below.

```
void Aged (int effect)
	{
		if(effect > 0)
		{
			if(age >= Ages.superOld)
				//Destroy (this.gameObject);
				//DestroyImmediate(this.gameObject);
				dead = true;
			else
				age++;
		}
		else if (effect < 0)
		{
			if(age <= Ages.pre)
				//Destroy (this.gameObject);
				dead = true;
			else
				age--;
		}

		HandleNewAge();
	}

void HandleNewAge()
	{
		if(dead == false)
		{
			switch(age)
			{
			case Ages.pre:
				transform.tag = "Acorn";
				foreach(GameObject img in Images)
				{
					img.SetActive(false);
				}
				transform.position = new Vector3(transform.position.x, acornYPos, transform.position.z);
				acorn.SetActive(true);
				treeCollider.radius = 0.1f;
				treeCollider.height = 0.22f;
				treeCollider.center = new Vector3(0,0,0);
				break;

			case Ages.baby:
				transform.tag = "Tree";
				foreach(GameObject img in Images)
				{
					img.SetActive(true);
					img.renderer.material = baby;
				}
				transform.position = new Vector3(transform.position.x, treeYPos, transform.position.z);
				acorn.SetActive(false);
				treeCollider.radius = 0.19f;
				treeCollider.height = 0.44f;
				treeCollider.center = new Vector3(0,-.89f,0);
				break;
			case Ages.young:
```

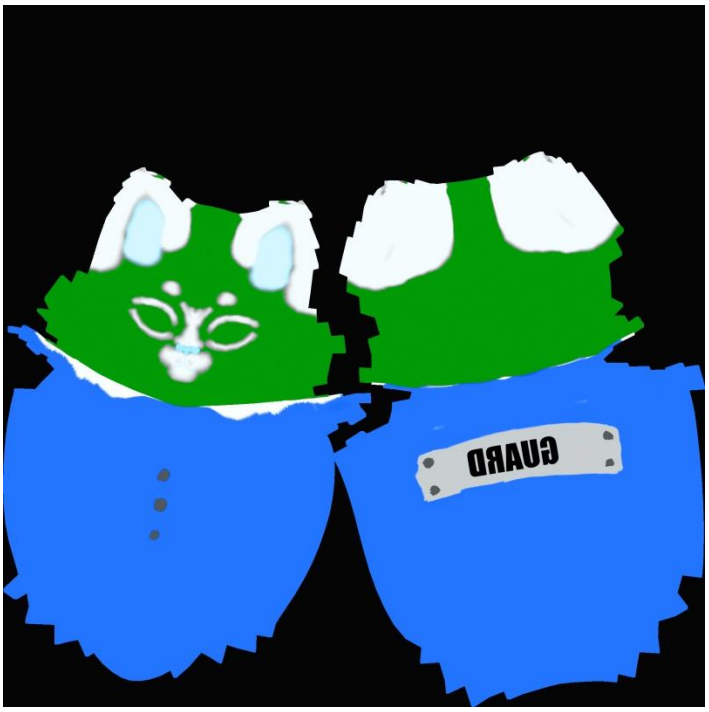
```
                                        transform.tag = "Tree";
                                        foreach(GameObject img in Images)
                                                img.renderer.material = young;
                                        treeCollider.radius = .35f;
                                        treeCollider.height = 0.78f;
                                        treeCollider.center = new  Vector3(0,-.76f,0);
                                        break;
                                case Ages.normal:
                                        transform.tag = "Tree";
                                        foreach(GameObject img in Images)
                                                img.renderer.material = normal;
                                        treeCollider.radius = .42f;
                                        treeCollider.height = 1.32f;
                                        treeCollider.center = new Vector3(0,-.41f,0);
                                        break;
                                case Ages.old:
                                        transform.tag = "Tree";
                                        foreach(GameObject img in Images)
                                                img.renderer.material = old;
                                        treeCollider.radius = .65f;
                                        treeCollider.height = 2.04f;
                                        treeCollider.center = new Vector3(0,0f,0);
                                        break;
                                case Ages.superOld:
                                        transform.tag = "Tree";
                                        foreach(GameObject img in Images)
                                                img.renderer.material = superOld;
                                        treeCollider.radius = 1.2f;
                                        treeCollider.height = 2.93f;
                                        treeCollider.center = new Vector3(0,0f,0);

                                        Vector3 acornSpawnPoint = transform.position + new Vector3(Random.Range(-1.2f,
1.2f), 0, Random.Range(-1.2f, 1.2f));
                                        Instantiate(treePrefab, acornSpawnPoint, treePrefab.transform.rotation );
                                        break;
                                default:
                                        transform.tag = "Acorn";
                                        foreach(GameObject img in Images)
                                        {
                                                img.SetActive(false);
                                        }
                                        transform.position = new Vector3(transform.position.x, acornYPos, transform.position.z);
                                        acorn.SetActive(true);
                                        treeCollider.radius = 0.1f;
                                        treeCollider.height = 0.22f;
                                        treeCollider.center = new Vector3(0,0,0);
                                        break;
                                }
                        }
                        if(dead)
                                Destroy (this.gameObject);
        }
```

Art Revision
Vinessa Mayer

Character texture for Kitty Break

Revised guard texture for Kitty Break



Art Revision

Patrick Ryan

Ground Sprite Version 1

The original ground sprite for the 2D Platformer prototype was mimicking the Mario Bros. level design where the character sprites sat on top of the ground sprites.

Ground Sprite Version 2

The revised ground sprite was designed to allow the character sprites to overlap the ground and simulate the character walking on a three dimensional ground. The character animation sprites took into account the distance between front and rear legs, so having them stand on top of a completely flat ground sprite would leave their rear foot hovering in the air. The height of the dirt was also increased to allow the camera to clip the bottom so that camera shake would not show the edge of the sprite.

Art Revision

Tyler Niemi

Monster Sprite Version 1

Monster Sprite Version 2

Art Revision

Shaquille Turner

Dragon Version 1

Dragon Version 2