
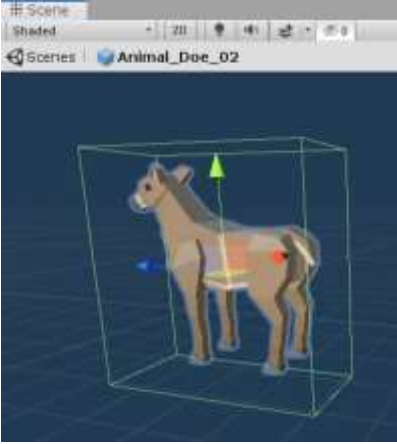


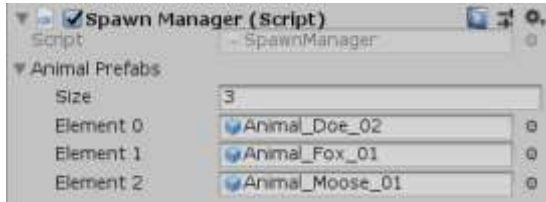
CORNELL NOTES – COMPUTER PROGRAMMING & GAME DESIGN I

	Topic/Objective: Level 8: Basic Gameplay	Name:
		Class/Period:
		Date:

Level Objective:
To create a top-down game using in-then statements, random value generator, arrays, collision detection, prefabs, and instantiation.

Questions:	Notes/Answers/Definitions/Examples/Sentences:
	<p>Prefabs</p> <ul style="list-style-type: none"> • What are they? – A prefabricated version of the asset you are using • Purpose – To use the asset anytime and anywhere in our scene • You know the assets in your Hierarchy Window are Prefabs when you see a blue cube next to its name • To open the Prefab in an isolated view from your other assets, double-click it in the Prefabs folder 
	<p>Input Manager</p> <ul style="list-style-type: none"> • What is it? – keeps track of when a player presses keys, clicks the mouse, presses a button on a controller, etc. during the game • If you aren't sure what key to use, use KeyCode in your command
	<p>Duplicating Projectiles While Playing the Game</p> <ul style="list-style-type: none"> • By using a concept called Instantiate • What is it? - Creating copies of objects that already exist during game play • This does not create new projectile Prefabs but rather uses projectile Prefabs that already exist
	<p>Removing Objects During Gameplay</p> <ul style="list-style-type: none"> • We do this using Destroy • What is it? - ability to remove an object, components, or assets from the game • Why do this? – help with memory in the game, remove items that go beyond the view of the camera, or to accept game-related events • Typically objects are not destroyed at the beginning when the game is played • Usually an if-statement is used to determine the condition of when an asset should be destroyed

CORNELL NOTES – COMPUTER PROGRAMMING & GAME DESIGN I

Questions:	Notes/Answers/Definitions/Examples/Sentences:														
	<p><u>What Does That Code Mean?</u></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 40%; padding: 5px;">horizontalInput</td> <td style="padding: 5px;"><u>Variable used to retrieve the input of the left and right arrow keys during play</u></td> </tr> <tr> <td style="padding: 5px;">transform.position</td> <td style="padding: 5px;"><u>Getting the players current position on the x, y, or z axis</u></td> </tr> <tr> <td style="padding: 5px;">Input.GetKeyDown</td> <td style="padding: 5px;"><u>Returns "true" when the player presses down on a certain key</u></td> </tr> <tr> <td style="padding: 5px;">Instantiate</td> <td style="padding: 5px;"><u>Clones the original object and returns the clone onto the scene</u></td> </tr> <tr> <td style="padding: 5px;">Destroy (gameObject)</td> <td style="padding: 5px;"><u>Turns off the current gameObject the script is attached to when a condition occurs</u></td> </tr> <tr> <td style="padding: 5px;">Random.Range</td> <td style="padding: 5px;"><u>Generates a random number in a certain range</u></td> </tr> <tr> <td style="padding: 5px;">InvokeRepeating ()</td> <td style="padding: 5px;"><u>Takes a method you want to call at a certain time and repeats it</u></td> </tr> </table> <p><u>Spawning</u></p> <ul style="list-style-type: none"> • Another word for spawning is <u>creating</u> • A <u>SpawnManager</u> script is created to add <u>location, timing, etc.</u> to the spawning of new gameObjects into the scene • To store all of these gameObjects you want to spawn into the scene, an <u>array</u> is used • In your script, two <u>brackets []</u> side by side mean an array has been created • Think of arrays as various shelves on a bookshelf holding onto various books • Remember: when you know the size of your array, the first element always starts off with a <u>zero</u> • When adding gameObjects you want spawned into your scene, remember to drag the <u>Prefab</u> into the <u>SpawnManager component</u>, not the original  <p><u>Local vs. Global Variables</u></p> <ul style="list-style-type: none"> • Where you put your variables in your script matters a lot! • Variables you want to use anywhere in your code are called <u>global variables</u> • Variables used inside particular methods only (ex. Update ()) are called <u>local variables</u> and CANNOT be used anywhere in your code <p><u>Collider and Trigger Components</u></p> <ul style="list-style-type: none"> • Colliders react to <u>collisions and forces</u> applied from a script • It's important to checkmark <u>trigger</u> in the <u>Box Collider Component</u> in order to detect an object within a particular space • Colliders require a <u>Rigidbody Component</u> to detect collisions in our physics 	horizontalInput	<u>Variable used to retrieve the input of the left and right arrow keys during play</u>	transform.position	<u>Getting the players current position on the x, y, or z axis</u>	Input.GetKeyDown	<u>Returns "true" when the player presses down on a certain key</u>	Instantiate	<u>Clones the original object and returns the clone onto the scene</u>	Destroy (gameObject)	<u>Turns off the current gameObject the script is attached to when a condition occurs</u>	Random.Range	<u>Generates a random number in a certain range</u>	InvokeRepeating ()	<u>Takes a method you want to call at a certain time and repeats it</u>
horizontalInput	<u>Variable used to retrieve the input of the left and right arrow keys during play</u>														
transform.position	<u>Getting the players current position on the x, y, or z axis</u>														
Input.GetKeyDown	<u>Returns "true" when the player presses down on a certain key</u>														
Instantiate	<u>Clones the original object and returns the clone onto the scene</u>														
Destroy (gameObject)	<u>Turns off the current gameObject the script is attached to when a condition occurs</u>														
Random.Range	<u>Generates a random number in a certain range</u>														
InvokeRepeating ()	<u>Takes a method you want to call at a certain time and repeats it</u>														

