Lesson 1: Intro to the DBB World Editor

Time: 45 minutes

Prerequisite: None

Goals

Students will learn how to navigate the dot big bang World Editor, the primary tool used for building spaces and creating game logic. We will also introduce the resource sharing systems that allow students to search for and use objects and scripts made by other creators. Finally, we will explore some of the tools in the browser that will help us analyze and understand how a project is working.

Review and Preparation

Students should already have an account on dot big bang. If not, they can be easily created at dotbigbang.com/signup.

Target Skills

- Editing a Shared Game
- Bird's Eye Toggle while editing
- Moving the Camera and Player while editing
- Selecting Objects
- Focusing Objects
- Selected Entity Panel
- Transforms - Position

Vocabulary

- Dev Tools Console
- Transform
- Camera
- VoxelObject
- Entity

- Entity Search
- Opening the Dev Tools Console

Lesson Summary

Students will open an existing project that features a small island with palm trees surrounded by ocean. We will start by learning how to edit a shared project to make their own changeable copy of the project, then use the camera to learn how to navigate the 3D space and inspect the contents of the project. Then we will learn how to search for voxel objects, templates, and scripts and add each of them to the project.

Wrap-Up and Extension Activities

Students who finish the lesson should be encouraged to continue building the world and experimenting with all the existing content. This tends to be a very engaging feature, and it is fine for students to add so many things that they want to start over with an empty world when we begin scripting in lesson 3.

Lesson 2: Intro to the DBB Voxel Editor

Time: 45 minutes	Prerequisite: None

Goals

Students will learn to use the Voxel Editor, the tool for creating cube-based 3D models known as voxel objects.

Review and Preparation

Students should already have an account on dot big bang. If not, they can be easily created at dotbigbang.com/signup. As a warm-up activity, you may want to explore existing community voxel objects on dot big bang for inspiration.

Target Skills

- User the camera positioning tools
- Enable and adjust mirroring by axis and position
- Using reference images
- Simplifying for voxel art
- Finding created objects in your dot big bang profile
- Remixing objects

Vocabulary

- Orbit
- Pan
- Zoom
- Reference Image
- VoxelObject
- Color Hex Codes

Lesson Summary

Students will learn to use each of the tools of the Voxel Editor while building a treasure chest, which will later be scripted in lessons 3 through 5. They will learn to find reference art for a 3D model, and make choices based on the theme of a future game. After creating the shape and colors, they will learn to save and remix the object into multiple pieces so that they can be scripted independently later in a game.

Wrap-Up and Extension Activities

Finishing the chest early is an opportunity to create many more objects in the voxel editor. Animation tools are also available but not explicitly taught in the lesson, so students can flex their creativity by making as many new objects after the initial chest as time allows.

Lesson 3: Scripting Part 1

Time: 45-60 minutes	Prerequisite: Intro to the DBB World Editor or self-directed practice with the World Editor.

Goals

Students will learn the fundamentals of creating scripts that move voxel objects to create an animation, and learn how to create code that repeats until it meets a particular condition. We will introduce some types of variables, as well as real developer reference materials to start learning like the pros!

Review and Preparation

Students can either use the world that they built in the first lesson, and pieces of a treasure chest built in the second lesson, or start directly here using an empty world and voxel objects remixed in the first step.

Target Skills

- Running code once vs looping
- Commenting code to stop running it
- Manipulating transforms over time to create animations
- Conditionals
- Using public variables to manipulate values
- Creating and calling functions

Vocabulary

- Variable
- Parent
- Local and World Transform
- Increment
- Boolean
- Vector3
- If-Statement
- Condition

	● Local Variable ● Function ● Call

Lesson Summary

Students can use the treasure chest top and bottom made in the previous lesson, or remix existing voxel objects to make a chest animation. We will begin by changing the height of the chest, then changing it gradually over time until a condition is met. We discuss different ways to save data in variables, from primitive numbers to Vector3 class variables, and then use conditional code and Boolean variables to make repeating code stop. Finally, students will refactor code into a specifically named function for clarity.

Wrap-Up and Extension Activities

Students who finish this lesson early can move directly to Scripting Part 2, or be challenged to find ways to use their knowledge to animate other voxel objects. Creating a falling coconut is an excellent challenge that uses similar code to the chest with a reverse trajectory, and can help reinforce what each part of the code does through experimentation.

Lesson 4: Scripting Part 2

Time: 60-90 minutes

Prerequisite: Scripting Part 1

Goals

Students will learn how to make multiple scripts controlling distinct objects communicate with each other, and practice the patterns learned in part 1 to control a more complex movement - rotation. We will also introduce some habits that can make using the code editor and dev tools both more powerful and more efficient.

Review and Preparation

This lesson repeats many similar patterns from Scripting Part 1 and challenges students to figure out the code themselves before revealing answers based on their experience from the first scripting lesson. A great way to start would be just to walk through the code from lesson 1, discuss what each variable's purpose is, and what each line of code does.

Target Skills

- Understanding rotation axes
- Combining conditions with logical operators
- Referencing entities in code
- Handling potentially null values
- Broadcasting events between scripts
- Making flexible code based on inputs

Vocabulary

- Euler variable
- EntityRef
- null
- And (&&), Or (||), and Not (!)
- Function Parameters
- Function Returns

Lesson Summary

Students will make the chest lid rotate open, experimenting with rotating along the x, y, and z axes to choose the correct rotation, and learn to use Euler variables to represent rotations. Next they will learn to reference other entities in the game and handle the potential that they do not exist. After this we will learn to make the code more flexible using function parameters to change the behavior of the function based on inputs, and using function returns to stop code early.

Wrap-Up and Extension Activities

Students who finish this lesson early can move directly to Scripting Part 3, or be challenged to find ways to use their knowledge to animate other voxel objects. Two interesting topics for further self-directed research would be to look into how Euler variables use radians rather than degrees, but the dot big bang interface allows you to create one with degrees. Understanding radians can be enormously helpful for programming in graphical spaces in the future. Additionally, we alluded to the use of Quaternions for rotations, which is a much more professional workflow. This interactive page can be a great place to start for a motivated student.

Lesson 5: Scripting Part 3

Time: 90 minutes	Prerequisite: Scripting Part 1 & Scripting Part 2

Goals

Students will be challenged to complete more of the code on their own first, with checkable solutions, to continue reinforcing the previous lessons. They will also learn to use triggers and trigger events to make code respond to where players are, and to complete a game loop by allowing players to reset the game.

Review and Preparation

This lesson finalizes the work of the previous two by asking students to figure out how to do described tasks using their own code as a reference. Demonstrating how to do this before the lesson begins can help teach this method. You could make a door that swings open and essentially copy and adapt code from the opening chest lid as a demonstration, and have the whole group help you work through the challenge. Professional coding involves a lot of looking up existing solutions and figuring out how to adapt them to your specific goals.

Target Skills

- Creating scripts on your own
- Writing functions based on pseudocode
- Sending data with event broadcasts
- Using tags to organize entities

Vocabulary

- Collision & Colliders
- Trigger
- Tag
- Array
- For Loop

• Grouping entities in arrays • Using loops to apply code to all members of an array • Creating randomness in code	• The Math Library

Lesson Summary

Students will begin by creating a trigger, and exploring the events that will execute code when the player leaves or enters a trigger, as well as continuously while they are in the trigger. They will then use this to make dig spots that activate the treasure chest animation, then learn to duplicate this code and create multiple spots. They will then learn a technique for making a script that controls the whole game, and give the dig spots a tag so that they can all be referenced together in an array. They will learn to select a random member of the array with the math library, and apply code to all the dig spots using a loop. Finally, they will make a button whose animation states change based on collision with a player that can send a signal to reset the game, and allow players to continue guessing which spot as the

Wrap-Up and Extension Activities

With the skeleton of a minigame in place, students who have finished all the steps should be encouraged to find ways to make this more challenging and unique. The most straight-forward way to continue building the game is to design a much larger environment and hide the dig spots so that players explore your work as they hunt for the treasure. Other possibilities, like applying a negative penalty to an incorrect guess will require some research (look at the Trigger Hurt template for a code example) or creating a timer using this UI tutorial.