Last modified on 2012-06-15 05:08:49

Daniel Schenker

--- ¶ ¶

Work History:

=------=Prior to 04/19/2012=------=
-I created the top level of the world (the part with the mountains, grass, trees, water, etc)

-blueprint for world
-grass blowing in the wind
-painting all textures
-creating varying tree specifications
-smoothed all terrain as just enough to allow for fluid movement, without taking away from intentional jaggedness in the terrain
-chose appropriate art, skybox, water color, and fog colour/thickness in order to create a better illusion of a continous world, in order to escape "end of the Earth" visible effects

Edited first person controller

-tweaked movement values to allow for:

-controlled skidding
-sideways only wall jumping

-added a temporary rectangular prism to represent a gun, so that all that needs to be changed is the art, and none of the physics

Created multiple bullets

-BeamGunBullet: Shoots a blue energy orb that isn't affected by gravity
-BouncyBall: Shoots a white ball that is affected by gravity and is bouncy

Added debugging information

-Draw a red line from the contact point to the normal each time a collision occurs
-Draw a white wire frame sphere at collision points
-Draw a white line from the initial launch position of the bullet to the latest collision point
-Draw a blue line from the initial launch position of the bullet to the current position

Started framework for entire project using pseudocode

Started creating a HUD

Has a toggle function which allows the mouse to unlock from the center of the screen and show the cursor while holding down a button
The HUD is currently invisible

=------=04/19-20/2012=------=
Made a group with Craig Lacey

Discussed goals, learned from each other how each other's code works

Started using SVN for the first time

Have 3 locations set up for SVN: school, Geek Squad USB memory stick, home computer

Edited scr_nav c# script

Added additional scenario checking in order to prevent accidental bad set up on bots

If a bad set up occurs, the bots will default to chasing after the player

Added debugging information

-Draw a cyan line from the previous collision point to the latest collision point

Changed Gun - Bullet Relationship

Previously:

The gun told the bullet everything it should do

Now:

The gun now only controls the fireRate and bulletSpeedModifier
The Bullet now controls it's own bulletSpeed, killDistance and timeOut

Gun:

Fixed variables and functions to proper access (public, private)

Bullets:

Fixed variables and functions to proper access (public, private)
Renamed BeamGunBullet? to bullet_BeamOrb
Renamed BouncyBall? to bullet_BouncyBall

Game Manager:

Attempted to create a Game Manager starting with a gizmo section, but it didn't turn out well. Going to retry again later.

=------=04/21/2012=------=
Gizmos:

Figured out gizmo issue with needing multiple gizmo calls to get them to work

Fixing this also explained that the reason code beneath the gizmos wasn't working since after the first gizmo call, it would jump to the corresponding gizmo call in the OnDrawGizmos?() function

I fixed this by redoing the entire tag and layer system.

The tags and layer system needs to be organized still, but now everything works as intended, meaning only rearranging needs to be done

Debugging:

Resolved issue with %s statements inside Debug.Log(). Apparently Debug.Log() is more like std::cout in c++ so I just needed to switch it to + instead.

=------=04/26/2012=------=
SVN:

Craig and I stayed an hour after class to figure out SVN issues.

-We determined that the terrain data kept saying that it was deleted, when rather the pointer to the terrain data was simply not being updated whenever we moved files around, even inside Unity.

To work around this we created a backup of the terrain data itself, so that every time the issue occurs, we can simply add the terrain data into the project by naming it to whatever the supposed missing file is called, to trick Unity into thinking that nothing ever happened.

-We also determined the cause for my innability to update using SVN is because I updated the library and cache files at some point, therefore making updating impossible.

To work around this, I copied all my changes down (which were thankfully all in scripts), and created a new SVN checkout space, and then re-implemented the scripts back as they were prior to when I updated the wrong files.
Now Craig, SVN and I all have the same version of the project.

Gun:

I worked with Craig in figuring out how to change the value of a prefab when a new object was instantiated so that guns can have multiple types of bullets, just like in real life.

=------=04/27/2012=------=
Peer Revision:

Went over most of Craig's code, learned how it worked, fixed minor issues, cleaned it up, etc.

Gun:

Implemented bullet changing mechanic
Implemented bullet switching timer (which also allows for easy gun switching as well, but hasn't been implemented as there is only one gun currently)

Statistics:

Implement stat tracking

-Created a statTrackingOn boolean which should be moved to the GameManager? later, for easy toggling of all stat tracking, which I later plan to allow the user to toggle in an options menu.

This is especially useful for if the user's computer can not handle the extra calculations that need to be done by stat tracking.

=------=04/28/2012=------=
Statistics:

Implemented Additional Stat trackers in PlayerGUI.cs

GUI:

Revamped PlayerGUI.cs completely

-Instead of using a button to toggle "lockCursor" and "showCursor", these are now controlled by "hudVersion".
-Toggling hudVersion is now a button press instead of hold, and is now mapped to 'h' instead of 'ctrl'
-Layed out framework for going forward with the player GUI, all that needs to be done now is to simply add the features, as all of the ordering and logic should be good now.

=------=04/28/2012=------=
GUI:

Made mouse toggle have it's own seperate hotkey 'm' and HUD is now has its' own control using 'h'

HUD:

Added an actual visible HUD while playing that actively changes values whenever neccesary. (Using mock values for now, need to implement Getters and Setters)

Work Plans:

HUD:

Add two layers to the HUD

-First Layer(Bottom): Statistical data
-Second Layer(Top): Aesthetics layer

Add HUD toggle control so that the user can customize the HUD

-No HUD
-Bottom layer only
-Bottom and top layers
-Top layer only

Different HUD Types:

1 - key stuff only

score
bomb ammo
text mesh on gun for current clip

lives
hp

2 - everything

hud 1 plus:
stat tracking

3 - nothing

Gun:

Add amount of bullets left in clip display on top of gun(s)
Add a gun swapping effect done through rotating a gun out of view and back into view
Add zoom function

Platforming:

Add a mario style rotating fireball roller but horizontally spinning rather than vertically (like in Super Mario Galaxy)

Terrain:

Create a cavern passageway

-Entry point started
-I'm not sure how to go about doing this as there are many different ways to do so. More research needed.

Add pathway to spikey mountains

-Donkey Kong style "pallet walkways" (kind of like the old Star Wars games Kashyyk area)

Add drop down location in spikey mountains

-Tie this into Craig's loading interface which transitions to his scene(s)

Gameplay:

Create a section where tons of self-destructing robots chase the player on a narrow cliff, forcing the player to shoot the robots to the edge of the cliff where an electric fence will destroy the robots

Bullets:

Try to immitate Tribes: Ascend guns+bullets (can add more tweaks later)

-Spinfusor: Create a copy of BeamGunBullet? and add explosion + self-destruct upon collision (when white wire frame sphere is drawn), and make each shot require to be clicked, rather than constant fire
-Bolt Launcher / Grenade Launcher: Same as Spinfusor but with gravity and different particle effects
-Sniper Rifle: When zoomed in act as hit-scan or extremely fast projectile

When not zoomed in show red laser pointer
Should this be affected by gravity?

-Assault Rifle: Rapid Fire
-Sub Machine Gun: Burst Shot

Text Boxes:

Create 3D rectangular prisms with a few cuts in the sides to represent a cartoony text box that floats in the air in 3D space

Heli Buddy:

Create a helicopter type robot kind of like Tales from the Sonic series, but acts like fairy from Legend of Zelda series

Puzzle Door Lock:

Artwork: DoorToBigTreasureLockConceptPartA.jpg
Leads to cavern which comes out in the middle of a cliff, which leads to an invisible treasure chest like in Legend of Zelda, but surrounded by a circular orb shield

The orb shield (See Orb Shield) is taken down by lighting two torches on fire by shooting them (like Twilight Princess)

part A (red) is colour panels jumping+shooting (like Tales of Graces f mid-game dungeon intro puzzle)
part B (yellow) looks like it comes earlier, but actually dives down along side of cliff to underwater and resurfaces on beach, where shooting rotators powers this part (like kids spinny toys in playgrounds at Ikea)

coding for doors:

Door

----
bool isOpen = false;
bool isLocked = true;
bool isPowered = false;
bool somethingInTheWay = false;
bool redPower = false;
bool yellowPower = false;
----
bool OpenDoor?();*returns whether door could be opened or not based on other conditions*
*bool CloseDoor?();*returns whether door could be closed or not based on other conditions
bool DisengageLock?();*returns whether lock could be opened or not based on other conditions*
*bool EngageLock?();*returns whether lock could be closed or not based on other conditions
bool PowerUp?();*returns whether power could be turned on or not based on other conditions*
*bool PowerDown?();*returns whether power could be turned off or not based on other conditions
void DoorClear?();
void DoorBlock?();
void RedPowerOn?();
void RedPowerOff?();
void YellowPowerOn?();
void YellowPowerOff?();

Powerlines:

Use Color.Lerp to have power lines pulsate between to different colours such as Crimson Red to Bright Red every 3.4seconds for example

Coding for powerlines:

Powerline
----
bool isPowered = false;
string offColour = (0.5, 0.5, 0.5, 0.75);*grey is (0.5, 0.5, 0.5, 1) which is calculated through R, G, B, greyscale and transparency can be obtained with 0 greyscale (I think)*
string powerColour = black;
----
bool CheckIfOn?();
void PowerOn?();
void PowerOff?();
*bool SendPowerTo?();*returns whether power sent was received or not. if not, continue game regardless, but choose to either log in debug console, or display on screen to user that the power couldn't be sent anywhere but the line will remain on

Part A of Puzzle:

Artwork: DoorToBigTreasureLockConceptPartA.jpg
A numbered sequence is written on a plack (black tile in artwork) such as 14387256 where each number corresponds to the colours of the rainbow
The player must enter this command in the correct order by jumping on the 4 ground tiles and shooting the upper 4 wall tiles
If the user makes a mistake the input is reset back to the beginning
If the user enters the password correctly the corresponding (red) powerline turns on

Part B of Puzzle:

The user must shoot waterwheel type wheels in the correct order
Each time the user shoots a spinner it spins and a sound is played
If the user hits a spinner in the correct sequence a good sound is played
Else a bad sound is played and the player must input the sequence all over again
Once the user inputs the complete correct sequence a better good sound is played and the spinners start auto rotating by themselves while the corresponding (yellow) powerline turns on

Game Manager:

Add a game manager that can be used as hidden interface