Black & White 2
Vortex

Created By: Willy
Date: July 23 2018

Version	Changes	Date
0	Initial Release	July 23 2018

Table of Contents

Introduction

What is Black & White 2 (BW2) is missing? Well a lot of things, but I am focusing on one thing featured prominently in the original Black and White (BW) game. The feature I'm referring to is the vortex, a large portal which appears after winning each land. The player could throw any object into it, and that object would then come out of the vortex on the next land. Basically the vortex in BW allowed the player to move stuff from one land to the next. You could send food, wood, people, miracle seeds, rocks, trees, any mobile object. When used effectively the vortex is extremely helpful in completing BW.

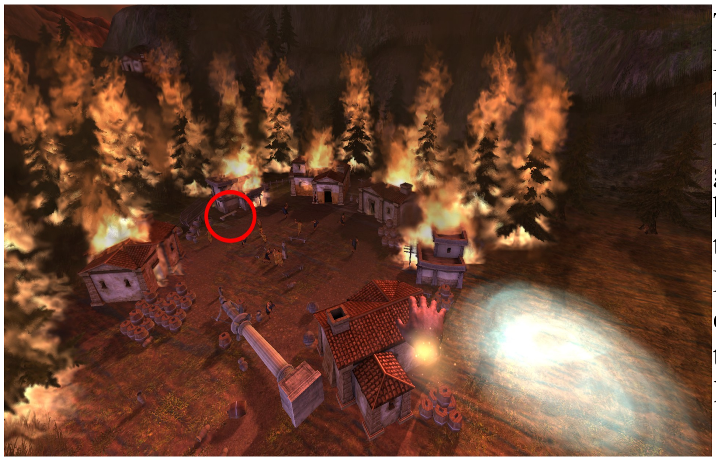
A big issue with BW2 is the fact that the vortex as seen in BW doesn't exist. There is a portal which appears when a land is complete. This portal like the vortex takes the player to the next land, but that is it. The player cannot throw objects into it and have that same object re-appear on the other side. The vortex in BW doesn't exists in BW2, in fact adding a vortex to BW2 was never the plan of Lion Head. No, instead the player was going to use ships to move objects to the next land. If you look at the original scripts for BW2 you can find code referring to this, for example in the Land1IslandDeparture.txt line 12:

```
say "It will ask you to check you have filled your boats for the trip..."
```

It is obvious then that they ran out of time and had to scrap this feature. Leaving BW2 without such an important feature, the vortex.

The purpose of this project was to find a way to re-create a vortex for BW2 capable of moving objects to the next land. At first the prospect of accomplishing this seemed impossible, there was no obvious way to save information on one land and read it on the following. But when the original scripts were released thanks to Handsome_Matt of bwFiles, we discovered a way.

In order to solve the issue of sending data across lands, there was one obvious place to start. In BW2 during the second land (Greek homeland) the player has to save a group of people by placing them in a portal! Those people then appeared walking out of the portal on the following land. This was exactly what the vortex script needed to accomplish. But the programmers at lion head cheated, in their script they simply create 10 random people on the 3rd land. In their code the player places 10 people into the portal, they simply delete that person, their information isn't saved or remember on the next land. The scripts involved with moving from land 2 to land 3 seemed to be a dead end.



There is a script located in the file Land3Intro.txt called BenchCarryingDude. It turns out that this script is an unknown secret Easter-egg in BW2. In land 2 during the second group the player is attempting to save, there is a bench beside one of the houses, if the player throws that bench into the portal then this script BenchCarryingDude is run which creates an extra villager carrying the bench. The only way this is possible is if there is data being sent from land 2 to land 3! Looking through the scripts

three lines of code were found:
```
persistent data "SPESHUL_BENCH" == 1
set persistent data "SPESHUL_BENCH" to 1
set persistent data "SPESHUL_BENCH" to 0
```
It turns out that persistent data is global to all lands. This was the key to finally creating a working vortex.

Now this way of saving data is quite limiting, and we imagine that this was not the way they were planning on implementing the fleet of ships. The commands use strings to refer to the data, but the scripting language does not provide anyway to edit strings. There is no way to dynamically add data, which means everything has to be hard coded. A vortex script like this one is much more limited then that of the one in BW. But it is possible, and now has been done.

The scripts provided contain the code required for a vortex in BW2. The files provided are not actual BW2 challenge files, you cannot take these files copy them somewhere and have vortexes appear in BW2. These files are script files which other modelers of BW2 can take and use in their own lands, I will leave it to some other brave soul to add these scripts to the original BW2 lands. This documentation is here for those who want to use these scripts on their lands, it will explain how to work with this vortex, what its limitations are, and what bugs I know of.

Features

This section explains the limitations of the vortex, or more specifically what objects it will accept and re-create on the other side, and what objects it does not. There are two types of vortexes:

1. Exiting - The vortex responsible for leaving the land, looks for objects around it, if one is found save the needed data and delete it.
2. Entering - The vortex responsible for entering a land, takes the objects saved by the exiting vortex and re-creates them.

Looking at all the objects in BW2 we can split them up into three categories:

1. Accepted - The exiting vortex can read the required data about the object and delete it. That object will be successfully re-created by the entering vortex on the next land.
2. Unaccepted - The exiting vortex cannot find or read the data needed about the object for re-creation. Or I was too lazy or found it unnecessary to code this object in.
3. Semi-Accepted - The exiting vortex can read required data about the object and delete it. But the entering vortex cannot re-created due to that object just not being coded in.

Lets look at some examples. lets say the player places a tree at the vortex, the code can identify that object as a tree, but for it to be re-created by the entering vortex we need to know what type of tree it is. To do that I use the command:
```
subType = variable get treeObj sub type
```
This returns a value from 0-10, if you look in the header file TreeInfoEnum.h found in the Script Compiler files you’ll see what type of tree the numbers refer to.
Knowing that this is a tree and what type of tree it is we can re-create it, and therefor a tree falls into the accepted category.

Now imagine the player threw a tree and the tree landed on its side, the game considers that tree to be of the type dead tree. The issue is that the sub type command will always return a value of 9999 no matter what type of tree it is. The code cannot read the sub type of dead trees! Therefore the code cannot gather the needed information to re-create. A dead tree must then fall under the category of unaccepted.

Imagine the player places a villager. This villager is an Egyptian Assassin (which is a possible type in BW2). The code will id this object as a villager, will be able to read the sub type as an Egyptian assassin. The villager will be deleted by the exiting vortex, but the information regarding this villager will not be saved as the code needed to re-create this villager doesn't exist for the entering vortex. This would be an example of a semi-accepted object. It will be deleted by the exiting vortex but will not be re-created by the entering.

The exiting vortex records the information needed for re-creation in a large array called data1. When the scroll is clicked the array is copied into the persistent data. The entering vortex then takes the persistent data and copies it into the array data1. The scripts then work directly with the array rather then consistently reading the persistent data. The reasoning behind this code design was to work around am issue within the persistent data, see section Known Bugs.

Now lets run through all the objects the vortex will accept.

Object Categories

Object	Accepted	Unaccepted	Semi-accepted
Trees			
Beech	🟩		
Spruce	🟩		
Cedar	🟩		
Oak	🟩		
Olive	🟩		
Pine	🟩		
Acacia	🟩		
Palm	🟩		
Big Palm	🟩		
Fir	🟩		
Maple	🟩		
Rocks			
Small Norse Rock	🟩		
Medium Norse Rock	🟩		
Large Norse Rock	🟩		
Rock	🟩		
Boulder Round	🟩		

Medium Boulder Round	🟩		
Large Boulder Round	🟩		
Boulder Small	🟩		
Boulder Big	🟩		
Boulder Tiny	🟩		
Boulder Huge	🟩		
Boulder Massive	🟩		
Elipsoid Small	🟩		
Elipsoid Big	🟩		
Elipsoid Tiny	🟩		
Elipsoid Huge	🟩		
Elipsoid Massive	🟩		
Sharp Small	🟩		
Sharp Big	🟩		
Sharp Tiny	🟩		
Sharp Huge	🟩		
Sharp Massive	🟩		
Eroded Small	🟩		
Eroded Big	🟩		
Eroded Tiny	🟩		
Eroded Huge	🟩		
Eroded Massive	🟩		
Block Small	🟩		
Block Big	🟩		
Block Tiny	🟩		
Block Huge	🟩		
Block Massive	🟩		
Siege Rock	🟩		
Eloi	🟩		
Moai Sad	🟩		
Moai Smiling	🟩		
Moai Funny	🟩		
Ore Rocks			
Small		🟥	
Medium		🟥	

Large		🟥	
Villager / Children			
Aztec Female	🟩		
Aztec Forester Male	🟩		
Aztec Fisherman Male	🟩		
Aztec Farmer Male	🟩		
Aztec Shepherd Male	🟩		
Aztec Leader Male	🟩		
Aztec Trader Male	🟩		
Japanese Female	🟩		
Japanese Forester Male	🟩		
Japanese Fisherman Male	🟩		
Japanese Farmer Male	🟩		
Japanese Shepherd Male	🟩		
Japanese Leader Male	🟩		
Japanese Trader Male	🟩		
Indigenous Female			🟥
Indigenous Forester Male			🟥
Indigenous Fisherman Male			🟥
Indigenous Farmer Male			🟥
Indigenous Shepherd Male			🟥
Indigenous Leader Male			🟥
Indigenous Trader Male			🟥
Greek Female	🟩		
Greek Forester Male	🟩		
Greek Fisherman Male	🟩		
Greek Farmer Male	🟩		
Greek Shepherd Male	🟩		
Greek Leader Male	🟩		
Greek Trader Male	🟩		
Norse Female	🟩		
Norse Forester Male	🟩		
Norse Fisherman Male	🟩		
Norse Farmer Male	🟩		
Norse Shepherd Male	🟩		

Norse Leader Male	🟩		
Norse Trader Male	🟩		
Isle Nymph			🟥
Sigved			🟥
Fenrick			🟥
Fenrick Stripped			🟥
Fain			🟥
Merry man 1			🟥
Merry man 2			🟥
Merry man 3			🟥
Merry man 4			🟥
Merry man 5			🟥
Merry man 6			🟥
Merry man 7			🟥
Merry man 8			🟥
Merry man 9			🟥
Merry man 10			🟥
Small Nose Boy			🟥
Robin Hood			🟥
Dimitrious acolyte			🟥
Dimitrious			🟥
Euronymous			🟥
Aztec Messagener			🟥
Aztec Coilbaron			🟥
Fell			🟥
Greek Jailor			🟥
Greek Boatmaster			🟥
Young Monk			🟥
Taranaga			🟥
Tezomoc			🟥
Hiroku			🟥
Prometheus			🟥
Inachus			🟥
Egyptian Assassin			🟥
Egyptian Slave Disguise			🟥

Sanura			🟥
Japanese Boat Master			🟥
Yuka			🟥
Old Monk			🟥
Black annis			🟥
Egyptian male A			🟥
Egyptian male B			🟥
Egyptian female A			🟥
Egyptian female B			🟥
Bazmet			🟥
Platoons - 60 Max			
Aztec Melee	🟩		
Aztec Ranged	🟩		
Japanese Melee	🟩		
Japanese Ranged	🟩		
Indigenous Melee	🟩		
Indigenous Ranged	🟩		
Greek Melee	🟩		
Greek Ranged	🟩		
Norse Melee	🟩		
Norse Ranged	🟩		
Aztec Melee	🟩		
Ghost Army	🟩		
Marauder			🟥
Seven samurai	🟩		
Monks			🟥
Aztec Melee Cinema			🟥
Aztec Ranged Cinema			🟥
Japanese Melee Reinforcements			🟥
Japanese Ranged Reinforcements			🟥
Norse Melee Reinforcements			🟥
Norse Ranged Reinforcements			🟥
Seven Samurai Reinforcements			🟥
Monks Reinforcements			🟥
Resources			

Food	🟩		
Wood	🟩		
Ore	🟩		
Miscellaneous			
Dead Tree		🟥	
Norse Statue			🟥
Norse Graveyard			🟥
Singing Stone Base		🟥	
Street Lantern	🟩		
Bonfire			🟥
Norse Tower			🟥
Norse Straight Wall			🟥
Norse 90 Wall			🟥
Teleport			🟥
Standalone alter			🟥
Toy Ball			🟥
Toy Teddy			🟥
Toy Doll			🟥
Tribute Box	🟩		
Norse Ship			🟥
Greek Ship		🟥	
Aztec Ship		🟥	
Japanese Ship		🟥	
Tombmarkers			🟥
Aztec Street Light	🟩		
Japanese Table	🟩		
Japanese Ladder	🟩		
Aztec Barrel 1	🟩		
Aztec Barrel 2	🟩		
Greek Barrel 1	🟩		
Greek Barrel 2	🟩		
Japanese Barrel 1	🟩		
Japanese Barrel 2	🟩		
Norse Barrel 1	🟩		
Norse Barrel 2	🟩		

Greek Urn 1	🟩		
Greek Urn 2	🟩		
Greek Urn 3	🟩		
Greek Urn 4	🟩		
Poo	🟩		
Nutoil Barrel	🟩		
Seven Samurai 1	🟩		
Seven Samurai 2	🟩		
Seven Samurai 3	🟩		
Seven Samurai 4	🟩		
Seven Samurai 5	🟩		
Seven Samurai 6	🟩		
Seven Samurai 7	🟩		
Greek Cart 1	🟩		
Greek Cart 2	🟩		
Norse Bench 1	🟩		
Norse Bench 2	🟩		
Greek Bench 1	🟩		
Greek Bench 2	🟩		
Japanese Bench 1	🟩		
Japanese Bench 2	🟩		
Norse Pot 1	🟩		
Norse Pot 2	🟩		
Norse Pot 3	🟩		
Aztec Pot 1	🟩		
Aztec Pot 2	🟩		
Aztec Pot 3	🟩		
Japanese Pot 1	🟩		
Japanese Pot 2	🟩		
Japanese Pot 3	🟩		
Greek Barrow	🟩		
Greek Wall 3 90	🟩		
Greek Wall 3 180	🟩		
Greek Wall 4 90	🟩		
Greek Wall 4 180	🟩		

Aztec Wall 1 90	🟩		
Aztec Wall 1 180	🟩		
Aztec Wall 4 90	🟩		
Aztec Wall 4 180	🟩		
Japanese Wall 1 90	🟩		
Japanese Wall 1 180	🟩		
Japanese Wall 4 90	🟩		
Japanese Wall 4 180	🟩		
Norse Wall 3 90	🟩		
Norse Wall 4 180	🟩		
Norse Wall 4 90	🟩		
Norse Wall 4 180	🟩		
Street Light Wood	🟩		
Street Light Norse	🟩		
Street Light Brass	🟩		
Street Light Japanese	🟩		
Norse Plant 1	🟩		
Norse Plant 2	🟩		
Greek Washing Line	🟩		
Greek Plant 1	🟩		
Plant Normal	🟩		
Palm Stump	🟩		
Palm Spike	🟩		
Palm Banana	🟩		
Hay Bale	🟩		
Siege Weapons - 20 Max			

Limitations

If an object is not mentioned above then it is not accepted by the exiting vortex. For example animals, no where above are animals mentioned. The original plan was to allow animals to be send through, but as it turns out the code can identify that there is an animal at the vortex but cannot read the sub type. Since the code cannot determine what type of animal it is, it cannot be recreated on the other end, and is therefore not accepted by the vortex.

If you are wondering about the ore rocks and why they are not accepted. It is because like the animals the code cannot read the sub type of the ore rock, therefore it cannot be recreated.

In order to be memory efficient much information has to be left out. For most objects the only thing recorded is how many of that object were thrown in. Information like size, scale, health, etc... are not recorded. When that object is recreated size and scale are set to 1, other attributes of an object are left as default. The recreations are not exactly the same as the object thrown in. On a more positive note, because only the amount of an object is recorded, you can throw as many of that object into the vortex as you want!

There is a limitation with trees in BW2, a tree can be placed in the ground or uprooted laying on its side. But for some strange reason the game identifies uprooted trees as dead trees. The code then cannot read the sub type of a dead tree, so dead trees cannot be recreated and therefore cannot be accepted. If the player want to send a tree through it must be place into the ground for the code to read it properly.

Some objects like armies and siege weapons required more information. For armies the number of men, type, and experience are needed. For siege weapons only experience was needed. Since more information was required for these objects the way the data is recorded is very different. The number of platoons you can send through is 60, the maximum number of siege weapons is 20.

Using the Script

In this section, I will explain how to use this vortex script. Like how to add the script into your land's code and how to use the interface I have created.

You may have noticed that there are two versions of the vortex script, a debug and a release version. There really isn't much difference between them other then the debug script will print out extra information for the player about the objects being thrown into the vortex. Release is meant for the final release of a landscape where all the debug stuff is removed and will never be displayed for the player. Although the disabling of the debug information can be easily done with the debug version.

Adding to Your Project

I will start by assuming that you have already setup a basic scripting project for your landscape. A basic scripting project contains these files:

1. ChallengeFile.txt - Containing a list of scripts for the compiler to compile
2. MainScript.txt - Contains a basic main script for loading the land like so:
   ```
   run script Main
   begin script Main
   start
           disable load screen
           set fade in time 1
           wait until 1 != 1
   end script Main
   ```
3. A compiler .dat file - This is the compiler for the project

These are the steps for adding the vortex script to your project.

1. Copy the vortex script into your project folder, it can be either debug or the release version
2. Open your challenge file and add the reference to the vortex script. This line must come before the file you are using to control the vortex.
   <Project Folder>/Vortex.txt
   Note: Make sure there is a space after the Vortex.txt

3. Compile the code make sure everything works
   The vortex script will add around 12000 instructions to the code, yes it is rather large.

There you have it the vortex script is now a part of your project!

Interface

There are a lot of scripts in the vortex file, but only a few of them you need to be know how to use. This is the complete interface of the vortex script:

1. Vtx_Create_Exiting(xPos, yPos, zPos)
2. Vtx_Create_Entering(xPos, yPos, zPos, playerTown)
3. Vtx_EnableDebug //Debug version only
4. Vtx_DisableDebug //Debug version only
5. Vtx_SetScrollType(Scoll_type)
6. Vtx_SetCloudAttributes(clGeneration, clHeight, clprecipitation, clOvercast)
7. Vtx_SetVortexAttributes(vtxSize, vtxColourChangeEnabled, vtxColour1_R, vtxColour1_G, \ vtxColour1_B, vtxHeight, vtxAngle)

8. vtx_created
9. vtx_debug
10. vtx_scroll_clicked
11. vtx_exporting
12. vtx_export_complete

Creating The Vortex
To create a vortex you need to call one of these two scripts:

1. Vtx_Create_Exiting
2. Vtx_Create_Entering

Exiting will create a vortex which the player can use to throw objects into. It is the vortex which will allow the player to depart the land and move to the next. The script will create three objects, (1) the vortex, a white swirling puddle looking thing. (2) a scroll above the vortex, can be bronze, silver, or gold. (3) a cloud far above the vortex, this cloud is used to create the lightening strikes. This script takes 3 parameters:

1. xPos : X position of the vortex
2. yPos : Y position of the vortex. The script will auto calculate the land height at the position provided so this value isn't very important
3. zPos : Z position of the vortex

Entering is the opposite of exiting, it is the vortex created when entering a new land. It will export all the objects the player threw in during the previous land. This script only creates a vortex, there is no need for a scroll or a cloud. This script takes 4 parameters:

1. xPos : X position of the vortex
2. yPos : Y position of the vortex. The script will auto calculate the land height at the position provided so this value isn't very important
3. zPos : Z position of the vortex
4. playerTown : The town the player starts with, needed when creating villagers, I need some sort of town to attach them to.

Examples:

run script Vtx_Create_Exiting(1300, 140, 1400) //Creates an exiting vortex at the location provided

run script Vtx_Create_Entering(1079, 140, 601, get town with id 0)
//Creates an entering vortex at the location provided and the town identified as 0

Both of these script will take care of running in the background the script needed for finding or exporting objects. Exiting runs two scripts

1. Vtx_Click_Check : Continuously checks if the scroll is clicked
2. Vtx_MainControl_Out : Looks for objects, if one is found saves the data and deletes it

Entering runs one script

1. Vtx_MainControl_In : Exports all the objects saved in the data

There are several variables used in these functions that you need to know about.

1. vtx_created : Set as true once the vortex is created and the scripts are running
2. vtx_scroll_clicked : Set as true when the scroll is clicked by the player
3. vtx_exporting : Set as true while the entering vortex is exporting the objects. It is set back to false when the exporting is complete
4. vtx_export_complete : Set as true once the entering vortex finishes exporting the objects

Cloud Settings
There is a script you can call to customize the cloud above the vortex.
This script is called: Vtx_SetCloudAttributes

It must be called before the creation of the vortex. It also takes 4 parameters:

1. clGeneration : Sets the speed which the could is generated
2. clHeight : How high up the cloud is above the vortex
3. clprecipitation : Amount of rain, 0 = no rain, 1 = as such rain as possible
4. clOvercast : sets the thickness of the cloud

Example:
run script Vtx_SetCloudAttributes(1, 300, 1, 1)
run script Vtx_Create_Exiting(1300, 140, 1400)

Vortex Settings
There is a script you can call to customize the vortex's settings.
This script is called: Vtx_SetVortexAttributes

It must be called before the creation of the vortex. It also takes 7 parameters:

1. vtxSize : Sets the scaling of the vortex visual effect, default is 2
2. vtxColourChangeEnabled : Enables custom colouring for the vortex, either 1 or 0
3. vtxColour1_R - vtxColour1_B : Sets the RGB colour of the vortex, the parameter vtxColourChangeEnabled must be equal to 1 for these to have any effect
4. vtxHeight : How far off the ground the vortex will be
5. vtxAngle : Vortex angle

The angle is only taken into account for an entering vortex. People and armies will head out of the vortex at this angle. Trees and other objects will be thrown in the opposite direct for the most part. It was coded this way to stop rocks and other objects from crushing your people and armies.

If you enter a 0 for any parameters then it will not be edited, unless that parameters is part of the colour change.

Example:

```
//Will only set the colour to disabled and the angle to -90, size and height are ignored
run script Vtx_SetVortexAttributes(0,0,0,0,0,0,-90)
run script Vtx_Create_Entering(1079, 140, 601, get town with id 0)
```

Scroll Setting

The script Vtx_SetScrollType will allow you to change the type of scroll that appears above an exiting vortex. Takes one parameter a value from 1-3:

1 - Bronze scroll
2 - Silver
3 - Gold

It must be called before the creation of the vortex.

Example:

```
run script Vtx_SetScrollType(1) //Makes scroll bronze
run script Vtx_Create_Exiting(1300, 140, 1400)
```

Moving to the Next Land

The vortex script doesn’t actual contain the code needed to load the next land it is up to you to code that using the interface provided by the vortex. After creating an exiting vortex you have to check the global variable vtx_scroll_clicked. Once that variable is equal to 1 that means the scroll was click and you can load the next land.

Here is an example:

```
run script Vtx_SetCloudAttributes(1, 300, 1, 1)
run script Vtx_Create_Exiting(1300, 140, 1400)

wait until (vtx_scroll_clicked == 1)
load map "./Data/Landscape/BW2/Land3549.bwe"
```

The vortex script will take care of saving the data before setting the value of vtx_scroll_clicked to 1.

Debug Version

The debug version has two extra scripts in the interface.

1. Vtx_EnableDebug
2. Vtx_DisableDebug

These scripts will allow you to display all the debug information whenever you want during runtime. You simply need to call the scripts.

Example

```
run script Vtx_EnableDebug
run script Vtx_Create_Exiting(1300, 140, 1400)
```

Known Bugs

At the current moment there is only 1 known bug. This bug was discovered while coding and attempted to be fixed but is still a potential issue you may come across.

Global Bug

This bug has to do with the internal workings of the persistent data. We found during testing that persistent data is global across all lands, saves, and profiles! In other words the set of data where the information is stored is the same for all lands, saves, and profiles. This has been deemed a very serious issue that we cannot over come. It has to do with the way lion head coded the persistent data.

Let me give you an example of how a player could take advantage of this bug. Lets say the player is playing through a series of 4 lands. Each land has the vortex script and uses it. If the player places lots of objects into the vortex on land 2, clicks the scroll going to land 3. Before the vortex begins exporting on land 3 the player exits to the main menu and loads a save on land 4, then presses restart land. Land 4 would then load in all the objects the player through in the vortex on land 2! because that was the last set of data saved to the persistent data.

The same issue exists between different players. If player 1 departs a land the persistent data would be over written, and if player 2 restarts their land then they'd get all the stuff player 1 had placed in the vortex in their game.

The first version of the vortex project was designed where it simply wrote directly to the persistent data, for example if you placed a tree by the vortex the script would just update the persistent data directly. When this bug was discovered, the project was very nearly scrapped. This bug was truly horrible for the first version. Imagine you beat a land and the vortex appeared, you began placing everything you needed for the next land into the vortex. Before leaving the land you had to leave your computer and do something else, you saved the game and exited. While you were gone your brother got on and loaded his profile and began playing, if the vortex was to appear for him it would overwrite all the objects you placed in the vortex in your game!

You can see why this project was nearly scrapped due to the globalization of persistent data. But we came up with a partial solution. The biggest issue with the first version is that the data could be overwritten at any point during the vortex process. To fix this the code was rewritten to record everything in an array. Then when the scroll is click the array is copied into the persistent data. The next land then copies the persistent data straight into the array and all the exporting and object recreation works from the array. This code design means the data is only vulnerable after the scroll is click and before the next land reads the data into the array. Thus the only way the player can get his data overwritten and get the wrong objects is if (1) They loaded a save where the game hadn't copied the data into the array yet, (2) when restarting the land.

The probability of the player creating a save before the data is copied is extremely low once you take in a count the need for a land intro cut scene.

There is still the possibility that a player could get the wrong objects due to it being overwritten, but that risk (Thanks to the array) is has been deemed acceptable.

Conclusion

There you have it, everything you need to know about how to use this vortex. I hope you don’t run into any issues.

If you do find any issues or have and suggestions for things I should change feel free to contact me on bwFiles:
http://www.bwfiles.com/forum/index.php?action=profile

or message me on discord.