REVEREX:DX Tutorialization Systems and TAFFY

COSC-3P99 - Final Technical Report

Supervisor: Prof. Renata Dividino

Thomas Berner
Department of Computer Science
Brock University
St. Catharines, Ontario
tb21kj@brocku.ca | 7240831

Abstract - REVEREX:DX contains many complex actions and rules of the world that players need to understand in order to have a successful and enjoyable playthrough. It is the job of the developer to communicate these rules and actions needed in a simple to understand manner. Players should be able to quickly grasp concepts and act reactively based on what is communicated in order to build an understanding of their actions within the scope of the world they are placed in. Throughout this report, we touch on methods and practices used in REVEREX:DX for communicating effectively with players and the tools that were developed to accomplish that goal.

I. Introduction

Initially, the focus of this report and my goal on the project was to touch on utilizing shader caching and optimizing game performance, but as the project progressed my focus shifted away from this original goal. As development continued, we felt there were holes in other areas where much more focus was needed, and one key piece was the tutorial. I was tasked with programming the elements needed to tutorialize the game as well as much of the design for the first level. This heavily drew away from shader caching and achievements development which was stated previously and those tasks were assigned to others later in development. This report will focus on the elements contained within the tutorial of REVEREX:DX, as well as the systems used to accomplish the design goals and philosophy the level is based around, such as the TAFFY system.

II. Background

REVEREX:DX is a two player cooperative game in which players take the role of medical technicians who operate an experimental machine, aptly named the REVEREX. The REVEREX is

used to help bring a comatose patient back from a comatose state by entering the deep subconscious mind and bringing them back to reality. One player will play as the Navigator, who will use quick gravity defying platforming to traverse the inside of the mind. The other will play as the Vitalist, who will play short microgames to administer small doses of medication, affecting the experience of the Navigator by speeding them up, making them jump higher, or prevent them from having a heart attack. The two players must work together to reach the final area of the mind and save the patient, before the timer runs out and the experimental elements of the REVEREX machine kill the patient.

As there are two players which need to be taught two completely different rulesets, but also have a basic understanding of the other players goals and gameplay, there is a complex balance that must be struck when creating the tutorial. In the tutorial, both players must be kept engaged no matter which player the tutorial is focusing on and must retain all information to a basic enough level to progress throughout the game without major struggle. The player who is the navigator should be able to look, run, jump, and traverse gravity defying areas without a major problem, outside of skill based platforming intended as a challenge within the level. They should also understand what information to communicate to the vitalist, such as their BMP or wanting a jump or speed boost in certain situations. The player who is the vitalist should understand which selectable microgame creates a given buff for the navigator and its effect. They should also know how to play each microgame that they are prompted with and quickly be able to complete it.

III. Methodology

In order to communicate with the players in game, we make use of the in game character TAFFY, also known as the Technical Assistant For Finding You. This maintains a level of diegesis to the tutorial and maintains immersion for the players. TAFFY takes inspiration from internet assistants from the 90's such as Clippy, the windows assistant, and acts as the key character who leads players through the REVEREX world. As TAFFY mimics one of these assistants, they tend to be very much in the face of the player, obnoxiously popping up and spewing out information on either player's screen with a large speech bubble and robotic voice. By having TAFFY be a tutorialization tool we can shove as much information in front of the players in a quick manner where they are almost forced to take it in with the character speaking and a speech bubble on the screen.

When creating the TAFFY system, it was key for it to be flexible, as if our design for the tutorial were to change or should we want to iterate upon it, that could require a very large rewrite. In order to accomplish this, the system is in two parts, the TAFFY manager [1], which holds all the functionality to make TAFFY function, and the prompt script [3], which is a component that can be placed on any volume in the world and feed information held within it to the

manager to trigger TAFFY. In addition to these there is a dat file which contains all the text for taffy to display on screen and a dictionary within the sound manager that contains all the voice files for TAFFY, which is indexed by the line number in the dat file to match sound to text easily.

The TAFFY manager functions by controlling a number of elements within the persistent scene. These include TAFFY's idle sprite beside the timer, the transition wires to both screens, and TAFFY themself, which contains the text and multiple images for different emotions and reactions. The manager contains a few other key elements including a list of integers for all lines that need to be read and spoken, and a few booleans that other scripts will need to reference to check states, such as if TAFFY is active, if the first room tutorial is active/complete, and what screen it should TAFFY be on or is on.

TAFFY uses the Raise and Lower functions to move between screens and interact with the REVEREX. The raise function is a simple coroutine, in which it checks which screen TAFFY should be on by checking the location boolean, if false move to the navigator screen, and if true move to the vitalist screen. It then lerps the light offset value of the wire between 0 and 1 to mimic the effect of TAFFY traveling through the wire. Once complete, it then pops up TAFFY onto the proper screen with the speech bubble empty and ready to display other text. The lower function is similar to the raise but just functions in reverse by disabling TAFFY first, then lerping the proper wire value from 1 to 0, then enabling TAFFY's idle sprite. The lower function also resets any variables that would need to be cleared for the next call, such as clearing the displayed string, setting the active boolean to false, and reset the taffy face animation.

The core of TAFFY's functionality comes from the PlayLines function which is the basis for all other TAFFY specific functions and is truly quite simple. It contains a for loop to run through all of the given integers in the list that need to be said by TAFFY, and plays the proper audio clip and displays the text to match. It then waits for X seconds where X is the length of the audio clip that is played before continuing with the next line.
These raise, play lines, and lower functions are contained in a larger function called FireTaffyLines which can be invoked over the network by prompts. This ensures TAFFY is synchronized between both player's devices and can be easily triggered should designers want to prompt TAFFY in a certain place.

The key to trigger these function is in the prompt scripts, which are placed around the level containing the following data:

1. list of integers matching lines that TAFFY should say in order
2. animation to play at the end of TAFFY speaking
3. boolean for if it is the first room
4. boolean to end the tutorial
5. enumerated type of the emote that TAFFY should display while speaking

When one of these prompts is hit by the player, the list of integers to say are added to the list in the manager, the prompt then invokes the FireTaffyLines in order to start up TAFFY at the proper time on both machines. It then changes the Emote integer in the TAFFY manager which will call a function attached to the public accessor which changes out the active TAFFY face sprite to the matching value. Should the prompt have the boolean for the first room checked, it would trigger a different method in the manager named FireTutorialLines.

FireTutorialLines is a different routine in the manager that is nearly identical to the FireTutorialLines but instead of PlayTaffyLines, there is a different function named PlayTutorialLines that replaces it. The key difference between it and PlayTaffyLines is that at the start of the function it calls another async function CheckForInputs to check the navigator player's controller to get basic movement and looking input. It will play the very first line voice line for the players informing them of what to do quickly, with a checklist on the screen of the vitalist with the goals they need to accomplish. The CheckForInput function contains an array of 5 booleans, the first and second are used as checks movement along the X axis, the first checks if they've moved in the positive direction (right) and the second negative (left). The third and fourth booleans are used to check the Y axis movement, the third positive (forward), and fourth negative (backwards). The fifth boolean is used to check the C-stick magnitude, which makes sure the player has looked around in the world. To ensure players don't accidentally complete the tutorial without really paying attention, they must move either joystick over the halfway point to count as a completion. When checking for task completion, we only check if players moved on a given axis for the required amount; a player could only walk forwards without walking backwards and it would count as walking along the Y-axis. When this is complete, the function sets a boolean to true which then turns off the checklist and enables the Vitalist selectable menu, continuing onwards with the rest of the tutorial.

In addition to the TAFFY prompt script, there is also another named TAFFY Minigame [2], in which TAFFY can queue up minigames with an explanation in order to tutorialize the better for the vitalist player. The basic functionality such as passing over lines to say to the taffy manager, animations, first room, end tutorial, and emotion display are all still included, but there are 2 more key pieces, minigame type and minigame line. The minigame type is an enumerated type and others can simply select from the drop down menu which minigame they would like TAFFY to queue up. They can also select which line index they would like the minigame to play after. This also invokes a different method within the manager, FireMinigameLines, which contains the Raise and Lower function as well as the PlayMinigameLines function which takes the minigame and plays it at the given line index while taffy is speaking.

Now since REVEREX is a networked game, the minigame type and line need to be synchronized between both machines. With the limitations of synchronized variables and net routines, we cannot simply pass over a minigame object to the manager. To get around this, we take the minigame type enum and convert it to an integer, and we take minigame line index and multiply it by 10. We then add the two values together and assign that value to the synchronized variable in the manager named PromptMinigame. This way in the PlayMinigameLines function, we can break down the integer into both pieces by either dividing by 10 or getting modulo 10. Dividing by 10 gets us the line index, and getting the modulo 10 gives us the minigame type which can be converted back to its enumerated type and then that can be queued up for the vitalist.

IV. Results & Mistakes

The result of creating the TAFFY system like this was overall very successful when considering the initial goal for these elements which was to create a flexible and easy to use system that others can use and modify. With iteration of the tutorial and level, there were minimal large rewrites needed since many elements of the TAFFY system were created in a flexible manner to not be stagnant or fixed in a level. As we rewrote much of the TAFFY dialogue and changed the layout of the level, the system did not need many changes and new lines could be added seamlessly.

TAFFY became a great tool to help many player understand the game better in the beginning especially with the voice elements, as in many games player tend to skip past dialogue options but by making this character visibly move throughout the device and not break immersion with a generic popup telling the player what to do, we found more player succeeded at completing the game.

One element that was a problem was using line numbers in the prompts, as many other designers and artists were confused with labeling audio clips with the proper line number and the same in the lines document. Should I get the chance to go back and rework the system, perhaps creating a string based key system in its place so it could be more memorable than a load of numbers. Another element is that the TAFFY systems are very coroutine dependent, making it not the most optimized system.

V. Summary

Overall the TAFFY system is a complex system which led to the tutorial's overall success. Others could use the system properly without major issues or complaints coming forward. The overall project benefited from the inclusion of the system and as did players when completing the game. TAFFY created an overall greater experience for players narratively and gameplay wise by informing them of information that was easily digestible and straightforward.

Citations

[1] DprssdChckn, "TaffyManager.cs," *Reverex DX - Tutorial TAFFY*, GitHub. [Online]. Available: https://github.com/DprssdChckn/Reverex-DX---Tutorial-TAFFY/blob/main/Code%20Snippets/TaffyManager.cs. [Accessed: 01-Jan-2025].

[2] DprssdChckn, "TaffyMinigame.cs," *Reverex DX - Tutorial TAFFY*, GitHub. [Online]. Available: https://github.com/DprssdChckn/Reverex-DX---Tutorial-TAFFY/blob/main/Code%20Snippets/TaffyMinigame.cs. [Accessed: 01-Jan-2025].

[3] DprssdChckn, "TaffyPrompt.cs," *Reverex DX - Tutorial TAFFY*, GitHub. [Online]. Available: https://github.com/DprssdChckn/Reverex-DX---Tutorial-TAFFY/blob/main/Code%20Snippets/TaffyPrompt.cs. [Accessed: 01-Jan-2025].