Space Game Overview

Platform: Windows PC and Windows Tablets with Unity

Summary

Within the Space Game room, 3 to 6 players will enter the environment and play together. Each user will play the game through the utilization of their tablet's GUI controls and their own physical set of buttons, knobs, and/or switches. During the game, players will need to manipulate these controls when they receive instruction from another player, where the instructions will be provided on the other players' tablet.

Regarding hardware, each tablet will communicate with an Arduino Uno board for physical control handling.

Controls

A **Control** is an object with which the player will be able to interact. There will be two categories of Controls:

- **Physical Controls** are the hardware controls interfaced with the app through the Arduino board. All communication with the Arduino board will happen through a USB connection.
- **Digital Controls** will be displayed on the tablet.

Both Physical and Digital Inputs will have these two types:

- **Latching Buttons**: These can be read in a pressed (1) or unpressed state (0).
- **Knobs**: These can be positioned to send a value between 1 and 5 (possibly less).

Digital Inputs will have two other types:

- **Sliders**: These can be positioned to a value between 1 and 8.
- **Momentary Buttons**: These always maintain an OFF (0) position, unless touched by the user

Physical Inputs will have two other types:

- **Rocker Switches**: These latching switches can be turned On (0) and Off (1).
- **Joysticks (x2)**: Rapid movement of both joysticks will generate a value of 1. No movement will send a value of 0.

All controls will have a label displayed on them. This label will appear as soon as a round begins and will be unique amongst all other controls. The labels will change at the start of each round.

The digital controls will have random sets of buttons, switches, knobs and sliders.

The physical controls can be different from one tablet station to the next. An XML document will be referenced for the control types and the amount of controls (e.g. 4 switches and 2 knobs). The app will receive the value of each physical control through the Arduino Uno.

Starting a Game

All tablets (clients) and the Windows desktop (server) PC will be continuously running the application.

The Windows desktop PC will act as the master for the room, where it will the various gameplay states (e.g. Initialization, Start, End). Additionally, this PC will also handle the room's screen animations.

Player station tablets will display a *Ready* button, waiting for the player to tap the screen. When all players are ready, the session will start.

Playing the Game

A game will consist of a tunable number of rounds. Within each round, the players will need to solve as many instructions as possible within a limited, tunable amount of time per instruction:

- Each active seat tablet will display one instruction, describing an action to apply towards a control. The instruction will always mention a label and the value that needs to be set. For instance, "Turn *Button A* ON ", "Set *Slider B* to 4", or "*Switch C* OFF" are possibilities. The instruction will refer to a control label associated with another player's station.
- The player in charge of the corresponding control will need to apply the requested instruction within a given amount of time. When an instruction is completed, a new instruction will appear in place of the completed one.
- Each time an instruction is completed, the team will earn a point towards the round completion point goal (tunable).
- If the instruction is not completed within the instruction time, the team will accrue a ship damage point.
- If the team's accrued *damage* point count meets a predetermined ship damage count, the ship will be destroyed and the game will end.
- If the team's accrued *instruction* point count meets a predetermined round point goal, the round will end. Thereafter, the next round will begin or the game will terminate, notifying the players that the game has been won.

Players will occasionally encounter black hole instances within the game. If a black hole is entered, the tablet GUI's will be flipped upside-down. Players will have the opportunity to avoid the black hole event through the manipulation of joysticks during a visual/aural black hole warning.

End of the Game

At the end of the game, the tablets will become inactive until a "new game" command is sent from the Master PC.

Note: This signal can be sent at any point during the life cycle of the game, where it will reset the game.

Server Connection

The Windows desktop PC will receive inputs from the Bam Kazam site server. This PC will listen for and emit JSON objects described in the *Escape_the_Room_Digital_Game_API_Doc.pdf* document:

- Receive:
  - gameStart: this will start the game and include the amount of players for the session
  - gameStatus: the app will respond to this with the current game status (see currentGameStatus in *Escape_the_Room_Digital_Game_API_Doc.pdf*)
- Send:
  - currentGameStatus: the current status of the game (waiting, in_session, error)
  - gameStarted: to confirm the game has actually started
  - endGame: to announce that the game has ended and the final results (win, lose)
  - gameHeartbeat: will be sent every 60 seconds to indicate the app is running

Controls and Gameplay Setup through XML Files

Various aspects of Space Game are defined within XML documents, referenced by both Server and Client game applications. Gameplay items such as player-station physical control layout, control names, and rounds-per-game are set within these documents. Additionally, IP addresses for player stations and server communication are also stored within these files. Refer to the *Space Game Client Server XML* document for further details.