CUT THE POWER – INPUT

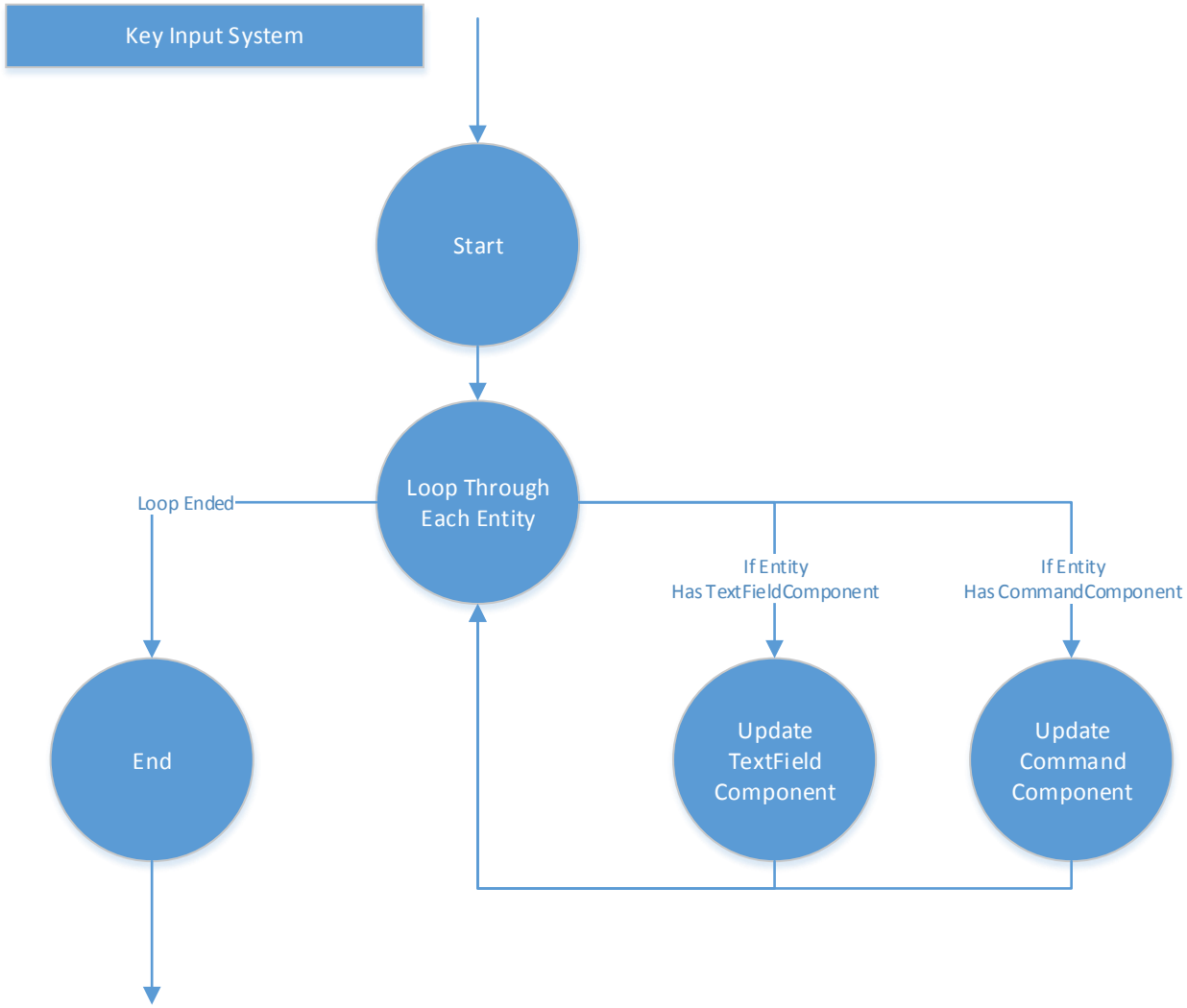
Cut the Power

Team Members:

Jordan Marling, Cory Thomas, Vincent Lau

1 TABLE OF CONTENTS

2 STATE TRANSITION DIAGRAM



3 PSEUDO-CODE

```
KeyInputSystem
{
    if (SDL_QUIT is triggered)
        close program
    set previous state of the keyboard to the current state
    update the current state of keyboard

    If a text field is focused
        add alphabetical keys pressed to the text value of the text field
    //end if
    else
        while looping through keys pressed
            if a command can be ascertained
                update command
            //end if
        //end while
    //end else
}

MouseInputSystem
{
    get new x and y positions

    set previous click states to current states

    while looping through entities
        update all entities with mouse components
        if entity is a text field
            if lclick previous state is down and current state is up AND click
happens in text field area
                set focus
                loop through other text fields and remove focus
            //end if
        //end if
        if entity is a button
            if lclick previous state is down and current state is up AND click
happens in button area
                set command component for button
            //end if
        //end if
    //end while
}

Add line to Chat
{
    if there is no text to add
        return

    add a new line to chatbox
    copy text to new line in chatbox

    if  last position is greater than chat length
        append new line to beginning


}
```

```
Chat Update
{
    clear the chat box
    set render modes

    for each indexed chat line
    {
        if (index is greater than the max # of lines)
            set index to first index

        if (index is the last line)
            break

        if (if text is empty)
            skip to next index

        set the alpha_percentage (transparency) to render based on how long
the text
            has been on the screen for

        if (if the alpha is less than 0)
            set the alpha to 0
        else if (alpha is greater than 1
            set alpha to 1

        flip alpha_percentage value
        set alpha based on alpha percentage
        draw the text

        if (alpha is not opaque)
        {
            set the pixels with new alpha
        }
    }

    clean up
}

Create Button
{
    assign as entity in world
    set components
        - menu item
        - rendered
        - position
        - button
        - mouse

    set position
    render text

    reset previous state
    reset current state
    reset hovered state

    set name
}
```

```
Create Label
{
    assign as entity in world
    set components
        - menu item
        - rendered
        - position

    render text

    set position in world
}

Create Title
{
    Create Label with Title Label ("CUT THE POWER")

    render with loaded TITLE font
}

Create Textfield
{
    assign as entity in world
    set components
        - menu item
        - rendered
        - positon
        - textfield
        - mouse

    if (it's a big text field)
    {
        load the BIG textfield image to the surface

        set size
        set position
        set max length
    }
    else
    {
        load the SMALL textfield image to the surface

        set size
        set position
        set max length
    }

    if (the entity's surface is empty)
    {
        failed to load the image error
    }

    if (text was provided)
    {
        add text to the textfield's text member
    }

}
```

```
Create logo screen
{
    assign entity in the world
    set components
        - menu item
        - rendered
        - position

    set position and size

    load image to surface
    if  (surface is null)
        error loading logo

    assign entity to world
    set component
        - menu item
        - rendered
        - position
        - animation

    set position and size

    load animation for the logo
    set id
    set render size
    start animation
}

Create keymap menu
{
    Create Labels for the remap-able commands
    load the keymap text file
    read the current key map from text file

    get the name the name of the key associated with the UP command
    create textfield for UP command
    get the name the name of the key associated with the DOWN command
    create textfield for DOWN command
    get the name the name of the key associated with the LEFT command
    create textfield for LEFT command
    get the name the name of the key associated with the RIGHT command
    create textfield for RIGHT command
    get the name the name of the key associated with the ACTION command
    create textfield for ACTION command

    create back,default and save buttons
}

Save keymap
{
    load the keymap text file
    write the keymap format with current keymappings
}

Created Animated button
{
    assign new entity to world
    set components
```

```
        - menu item
        - rendered
        - position
        - animation
        - button
        - mouse

    set position
    set size
    set render size

    load the animation
}

Animation System
{
    for (each entity)
    {
        if (animation playing)
        {
            adjust animation to frame rate


        }
    }
}

Load animation
{
    load a text file
    read in image names
    load images into appropriate frames

    get additional features from text file
}
```

4 MILESTONES

4.1 MILESTONE 1 – BASE SYSTEMS AND COMPONENTS

This goal for **February 21, 2014** will include 2 systems, a basic key input system and a mouse input system that updates the "Key Command" component and "Mouse" component for the player and menu entities. We will work on the setup menu from this stage onward and should go to the end of Milestone 2.

4.2 MILESTONE 2 – MENU

Due by **March 6, 2014.** During this stage, we will be finishing the initial setup menu with textbox inputs and buttons for modifiable connection parameters. After finishing that, we are helping out the Gameplay group, dedicating more resources to the gameplay systems.

4.3 MILESTONE 3 – MOUSE LOOK CONTROL AND KEY MAPPING

Due by **March 20, 2014.** For this final milestone, we will be working side-by-side with the Gameplay group to implement mouse looking and their final goals. On our own extra time (if applicable), we will be implementing a key mapping feature, using file parsing to attach keys to commands as opposed to hard-coding the relationship.

5 TEAM MEMBERS

5.1 JORDAN MARLING

- Designed and implemented the menu system
- (Re)designed and implemented the key input system
- Designed and implemented the mouse input system
- Designed and implemented the animation system
- Designed and implemented the cut-scene system
- Designed and implemented the key map system from file and menu
- Designed and implemented the sound system
- Designed and implemented the in-game chat
- Created the working base world for initial testing and integration
- Implemented text rendering
- Fixed text rendering issues
- Assisted everyone in the learning of the component system
- Redesigned the command component
- Designed score system (not used)
- Implemented the map loading
- Set the coding guidelines

5.2 CORY THOMAS

- Implemented menu system – key mapping
- Fixed key mapping in the menu
- Added functionality to textboxes
- Fixed bugs in the “Options” menu
- Redesigned “Options” menu for visual appeal
- Designed key input system
- Designed mouse input system
- Assisted in creating and designing entities

5.3 VINCENT LAU

- Designed key input system
- Designed mouse input system
- Troubleshoot menu issues
- Implemented text rendering
- Fixed text rendering issues
- Assisted in debugging Server test applications
- Input Documentation

6 TEAM REVIEW

As one of the smaller teams with the supposed least amount of work to do, we thought we would be moving between teams a lot more than we did. The input section of the game ended up taking on a lot more than we imagined which ended up with a lot less time to assist other teams. If we had researched more deeply into input systems, we might have prevented the loss of time before the first milestone. Jordan, the project leader, had many tasks on his plate such as helping the class learn the component system which took away a lot of time. He ended up producing a lot of progress than the rest of the Input team, being the most familiar with the underlying systems and concepts.

7 OBSTACLES

To elaborate on some issues we had while working on the input systems, we did not expect multiple commands per press of a key. This was fixed by introducing a previous state flag for the key press so that it would not induce an additional command.

Another issue that we encountered was with text rendering and visual appeal. To render text in one of the first phases of the menu system, the labels and buttons were all drawn as pictures. This proved to be an issue as the image “hitbox” was the same size of the image which was enlarged to fit every label or button. This meant that a simple “Play” button would have the same hit-box as the “Sound Off” button. This is when True-Type Font (TTF) rendering was designed and implemented. TTF rendering did not come painless either, we were never able to figure out the problem with a certain type and order of rendering the text. This obstacle was with the rendering using solid styles to have more rigid edges with the coloured text and the blended rendering of the outline.
We eventually settled for a more 3D look without colour bleeding through the outline using a different order of rendering and blended throughout.