Keeper's Call

Game Design Document

Minstrelcy Studios

J.R. Omahen

November 30, 2015

Contents

1

Game Design

1.1 Summary

A girl is wandering through the forest, playing near a lake. She hears a mysterious voice: a call, beckoning her to the water. There she discovers her true calling.

1.2 Gameplay

The player operates in first person, exploring the area surrounding the lake. The mysterious voice will beckon to the player, but the player won't know where it's coming from. The main obstacle to the player is a lack of familiarity with the environment, and not knowing where the voice is coming from.

1.3 Mindset

The player should be intrigued, seeking to discover what the voice is, where it's coming from, and who it belongs to.

2

Technical

2.1 Summary

The game is a text-based adventure game (*interactive fiction*), with very simple keyboard-based input. The target play time is **10—15 minutes**.

2.2 Screens

The opening screen will have an intro title, graphic, and note about the ”help” option. The word ”help” can be entered at any time to give the player the list of acceptable commands. The player will need to type ”Start” when they are ready to begin.

Each screen following will provide the player with a description of the area, including details about which directions are available to them. Finally, there will be a few areas where the player will ”hear something,” giving them the option to ”listen.” These scenes will be important story building points.

2.3 Controls

Players will be able to enter the following inputs:

Directional

- Straight

- Left
- Right
- Turn around

Actions

- Listen
- Don't Listen
- Pick up

Support Menu

- Help

2.4 Mechanics

The game has two basic mechanics:

Start -> Scene Description -> Direction input -> New Scene Description

Start -> Scene Description -> Action input -> Action Response -> Direction input -> New Scene Description

3

Level Design

Keeper's Call features a very simple environment with two thematic elements, and a simple story outline.

3.1 Thematic Elements

1. The Lake
    a) Location
        i. Scottish Loch
    b) Weather
        i. Overcast
    c) Ambience
        i. Opaque water
        ii. Cool, cold
        iii. Misty
2. The Forest
    a) Location
        i. Surrounds The Lake
    b) Ambience
        i. Dark, dim

ii. Rustling noises
iii. Wind or breeze can be heard

3.2 Story Flow

4

Development

Development will happen in two phases: underlying engine and game proper.

4.1 Fabler

Fabler is the name of the game engine itself. It is a simple graphically-rendered text engine for interactive fiction. It is written in JavaScript.

Architecture

Components drive the functionality of the engine internals. Each feature will be implemented as component object that can be plugged in to the engine framework. Additional features mean additional components, which can be tested individually.

The text will be rendered graphically onto a `Canvas`, and input taken by key strokes. Input will be collected by an input listener, then fed to a command parser. The parser will take the input and interpret the instructions. Any changes to the game environment will be requested through the engine by the parser.

Command parsing will be simple: instructions are one word, and can have one or more arguments afterwards. Each command will correspond to a function that will accept the arguments.

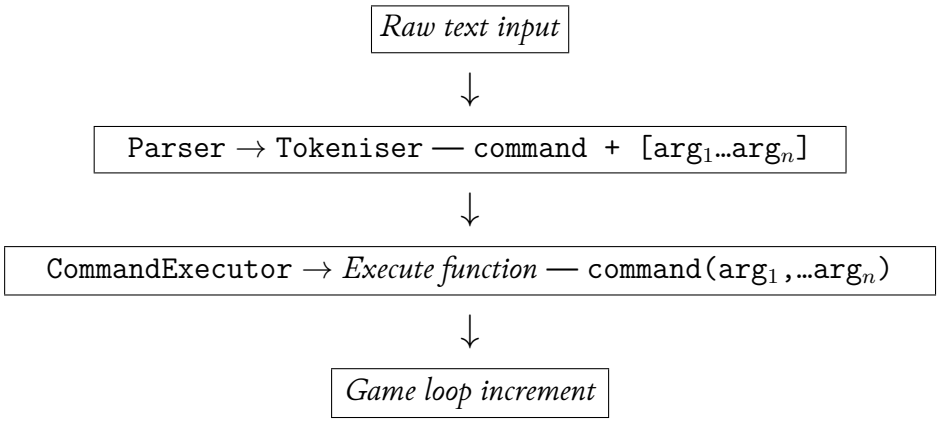
The game loop will be responsible for orchestrating the mechanics, input and display rendering. The game loop should be supplied by the game implementation, with a simple skeleton structure provided by Fabler.

Command Parser

There will be a simple command parser (outlined above), that will accept input in the following format:

`[command] + ␣ + ([argument`$_1$`] ␣ [argument`$_2$`] ␣ ... [argument`$_n$`])`

The processing of input looks similar to the following:



Dictionary

The proposed list of commands is as follows. In addition to commands and arguments, there will be necessary tokens to make a phrase or sentence grammatical, but is irrelevant to the command itself. These will be removed during parsing, but available to the end user.

Commands
go
look
take
open
read
talk
listen

Directions
forward
back
left
right

Tokens
a(n)
the
of
to
in
on
for
with
—(')s

Components

- Game (FABLER)
- Module

Component Compositions

- Module
  - GfxMan
  - Screen
  - Player
  - Location
  - World
  - Input
  - Parser
  - Event
  - Item

4.2 Game Program

Application Container

The game will run on the *Fabler* engine, encapsulated in an embedded *Chromium* instance, and will be written in JavaScript.

Startup

Upon launch, the game will run in full screen[1], presenting itself as on old terminal console.

[1] This may requiring prompting the user with a dialogue confirmation

Game Loop

The game loop is managed by *Fabler*, but the creation of all assets, story points and game flow will be hard–coded into the game program for simplicity. We will not be concerned with portability or ease of modification.

Shutdown

The game will exit by either the successful completion of the story, or the player entering an appropriate command in the terminal.

5

Development Timeline

Total project estimate is **30 hours**.

5.1 September 2015

- Game concept is agreed upon, and pre–production begins
- Brainstorming results in the majority of game design outlining
- GDD begun and developed

5.2 October 2015

- Pre–production grinds to a halt as team members are distracted with illness, work and other pursuits
- Some amount of tinkering with the GDD and server ensues

5.3 November 2015

- Pre–production resumes with the finalising of the GDD

5.4 December 2015

Starting with this month, development time is assumed to be **1-3 hours per week**.

- Production begins on the Fabler engine
- Production begins on the box an manual art
- Production begins on the story assets
- Some early testing of the running game engine begins

5.5 January 2016

- Testing begins of the game in the engine
- Final assets assembled into game
- Full game is tested and refined

5.6 February 2016

- Game is released to manufacturer
- Marketing material is created
- Manuals are created
- Final packaging completed
- Game published

6

Proposals

Please place proposals in new sections with any relevant and necessary information. If a proposal is adopted, ensure that it is added/incorporated into the relevant section. If discarded, please add it to the *Rejected Ideas* chapter, along with the discussion or reasoning.

6.1 Brainstorming Notes

These represent the areas agreed upon. Extra brainstorming notes available in the documents for Keeper's Call.

- Girl is the keeper of the Loch Ness monster
- Game is entitled *Keeper's Call*
- Nessie calls out. We want to avoid something inherent about the girl that makes Nessie want to reach out. We can allow readers to believe she is worthy, but gradually her flaws are revealed

7

Rejected Ideas

7.1 Hergen, Bergen, Fergen

With an option on *Dergen*.