INPUT PROCESSING

# Introduction

This document explains how all input devices are processed/handled/managed in the Boku framework. Currently, Boku supports gamepad, keyboard and mouse input; touch control, and multi-touch gestures in particular are to be added. The intent of this document is to layout how inputs are processed by the game in such a way as to clarify how a new input system may be functionally added in a modular fashion.

# BokuGame.cs

This is the main entry point of Boku. Input system initialization may occur in Initialize(); currently GamePadInput is the only input system requiring initialization. The main game loop is occurs in Update(); all input systems will require an update call here.

Boku/Common contains all the high level classes for GamePadInput, KeyboardInput, and MouseInput. Boku/Input contains all the input building blocks used by those aforementioned higher level classes.

Each frame, each of the input systems updates its own state and any game objects may then query any of the input systems for their current state to determine if any relevant input has occurred. There is no central “action interpreter” or event system which drives input – rather the onus falls to each game object to determine which input system it should be listening to, query the states relevant to itself in its own update loop, and determine what action needs to occur as a result. (i.e., pull/poll architechture as opposed to push).

Some lower level objects provide base input handling (mostly for gamepad and keyboard input). E.g., UIGrid has built-in handling for move left/right/up/down with the gamepad or keyboard (and mouse scroll wheel). Thus, a class like LoadLevelMenu which includes LoadLevelUIGrid derived from UIGrid does not need to implement these controls, and only requires mouse input handling to be defined (plus any control implementation specific to that screen).