**ABSTRACT**

In Microsoft’s Kodu Game Lab, children create games by writing behavioral rules for animated characters that inhabit 3D virtual worlds. Rules are composed of behavior-based robotics primitives such as “see” and “grab”, which suggests that Kodu might also be useful for programming actual mobile robots. But this requires overcoming the perception and manipulation disparities between Kodu characters and physical robots. While perception in a Kodu virtual world is omnidirectional, instantaneous, and error-free, and manipulation actions are effortless and perfectly reliable, none of these properties hold for robots in the real world.

We describe three strategies that were developed to reliably implement Kodu’s idealized primitives on a mobile robot. They are Navigation Error Monitoring, Fault-Tolerant Manipulation, and Perceptual Multiplexing. These strategies were implemented on the Calliope2SP, a mobile manipulation platform used for robotics education and research.

**METODOLOGY**

1. Analyzed how rule interpretation works in Kodu Game Lab.
2. Implemented a Kodu Rule Interpreter using Tekkotsu. (sample diagram to right)
3. Formalized how perception and manipulation work in Kodu.
4. Examined the perception and manipulation differences between Kodu characters and the Calliope2SP.

**THE PERCEPTUAL MULTIPLEXOR**

- Perceptual tasks:
  - Each implemented as a C++ class.
  - Uses vision to detect environment or robot’s state.
  - Dynamically created by running actions or evaluated conditions.
  - Stored within a queue for execution.

- Perceptual Multiplexor implements each strategy within one of two stages:
  1. Executes perceptual tasks using Round-Robin approach (perceptual multiplexing).
  2. Handles task failure recovery (navigation error monitoring and fault-tolerant manipulation).

**FAILURE RECOVERY**

- Halts and resumes rule interpretation.
- Localization recovery:
  - Handles localization and navigation error monitoring failures.
  - Goal: localizes through the use of a collection of April Tags strategically placed in the world.

- Object manipulation recovery:
  - Goal: recover the canister the robot lost or did not grab successfully.
  - Composed of several phases including two preparation phases.
  - The goal of the two preparation phases is to reposition the robot and locate lost canister depending on how the robot failed manipulation.

1. Robot lost canister while moving.
   - Reposition body: reverses to better detect lost canister.
   - Locate lost canister: searches for canister through incremental camera pans between current and last orientations.

2. Robot failed to successfully grab canister.
   - Locate lost canister: searches for lost canister near the gripper.
   - Reposition body: reverses body such that the gripper is approximately 150mm from canister.

**RESULTS**

- The Rule Interpreter successfully interpreted a Kodu program in which the robot reliably transported a red canister to a green canister.

- Canister transportation demonstrates the robot performing all three strategies.

- Project source code at: www.github.com/troiwilliams/tekkodu

**FUTURE WORK**

- Developing a multi-robot environment.
- Redesigning primitives to operate in a multi-robot environment.

**RESOURCES**

**KODU GAME LAB**

- Use rule-based programming to control Kodu characters.
- Makes programming perceptions and actions easy.

**THE CALLIOPE2SP ROBOT**

- Educational robot designed by RoPro Design, Inc. with CMU.
- Approx. 1 foot in diameter and 1.5 feet high.

**TEKKOTSU**

- Free, open source project developed at CMU.
- Enables high-level programming on multiple, predefined robotic platforms.
- Provides robotic capabilities such as vision, manipulation, and navigation via the Crew: MapBuilder, Grasper, and Pilot.

**REFERENCES**
