Model Based Testing of a Game Engine

using a Mono/.NET port of GraphWalker

UCAAT 2014
16-18 September, Munich

Marek Turski, Unity Technologies

About Unity

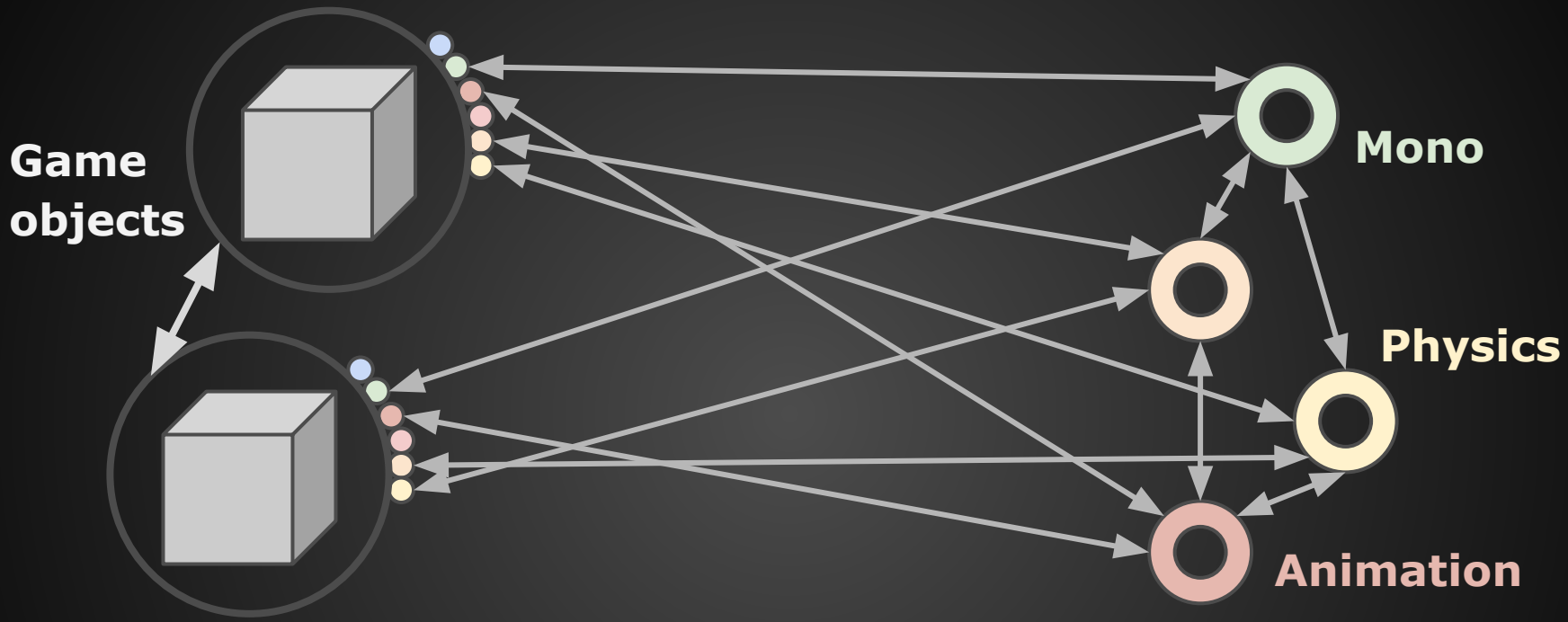
Ori and the Blind Forest - an upcoming adventure game created in Unity

Integrated development environment for creating games and other interactive virtual content (Windows/Mac OS X)

- Over 20 supported runtime platforms
- 100 core product developers, 15 test developers
- User base: 3.3 million registered developers
- End user reach: 600 million people

unity

Game engine complexity



Game engine software testing is complex!

- Thousands of game objects in a virtual environment ...
- Reacting to input from a system of interacting sub-engines ...
- Through interfaces called Components ...
  - Embedded Mono scripting framework interface
- Executed in fame-based fashion in real-time

unity

Unity QA Challenges

Automation

- Teams: Test Developers and Test Framework Developers
- **Large scope of automation frameworks**
  - Unit, Integration, Runtime, Scene-based, Model-based, Performance, Graphics, Import, …

← over 3000 automated tests!

Manual testing

- Teams: Student Workers and Test Engineers
- Integrated bug reporting system, dedicated user test groups
- Continuous functional, usability and regression testing
- Regular exploratory and release testing

Challenges

- Large test domain and fast development pace
- Low reuse of test artifacts from manual testing in automation
- Automation focused testing on unit-level functionality

Model Based Testing at Unity

Tools are expected to be robust - robustness requires high-level functional and integration testing!

Model Based Testing (MBT)

- Flexible test scope and execution parameters
  - Better product exploration, tests retaining value over time
- Model as a test artifact
  - Easier maintenance, additional source of documentation

Spec Explorer

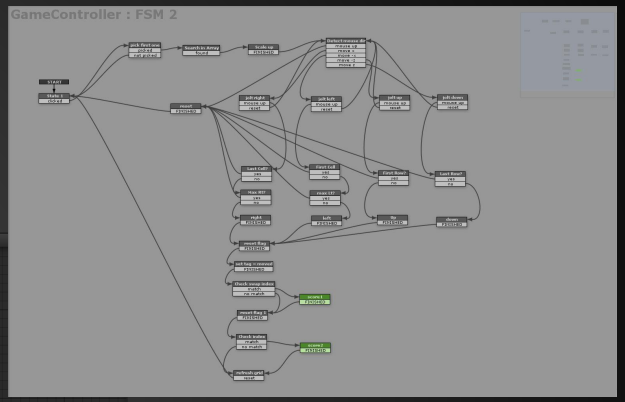
- Dedicated modeling language and exploration workflow
- Conformance testing of system state in a slice of the model

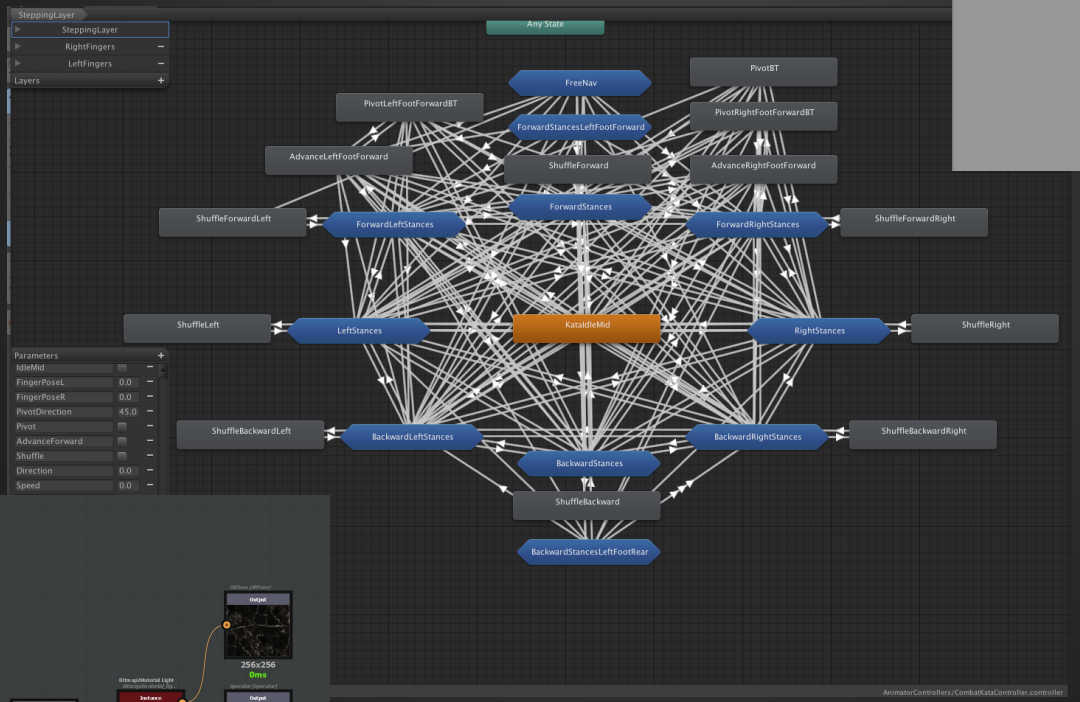
GraphWalker

- Lightweight workflow focused on visual model design
- Run-time binding with an implementation class

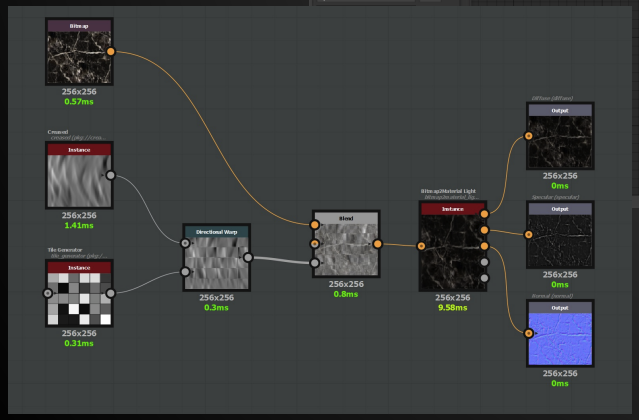
We’re good at creating models!

character AI behaviour graph →





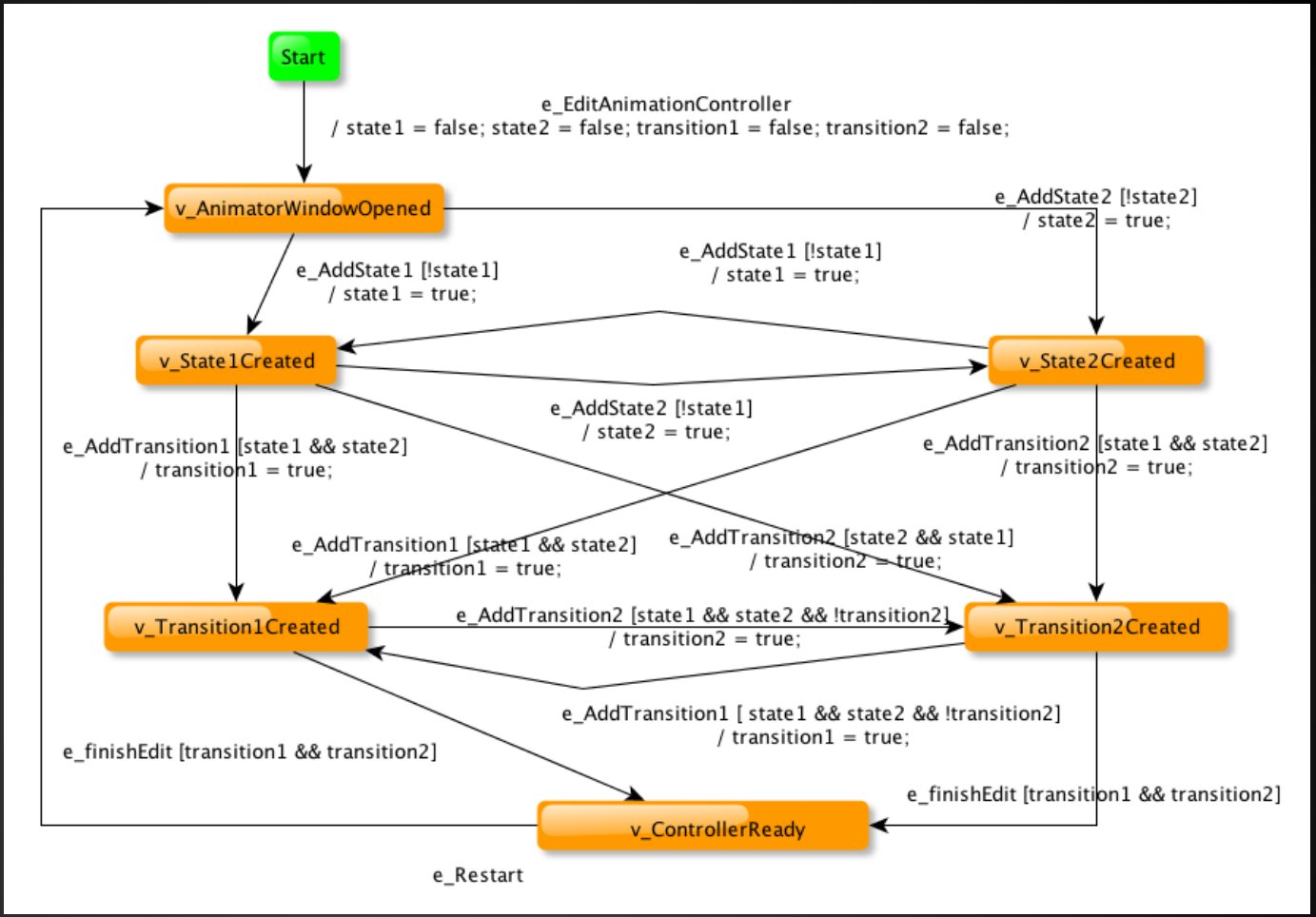
← animation graph



← procedural texture generation graph

unity

yEd Model
6 states
14 transitions

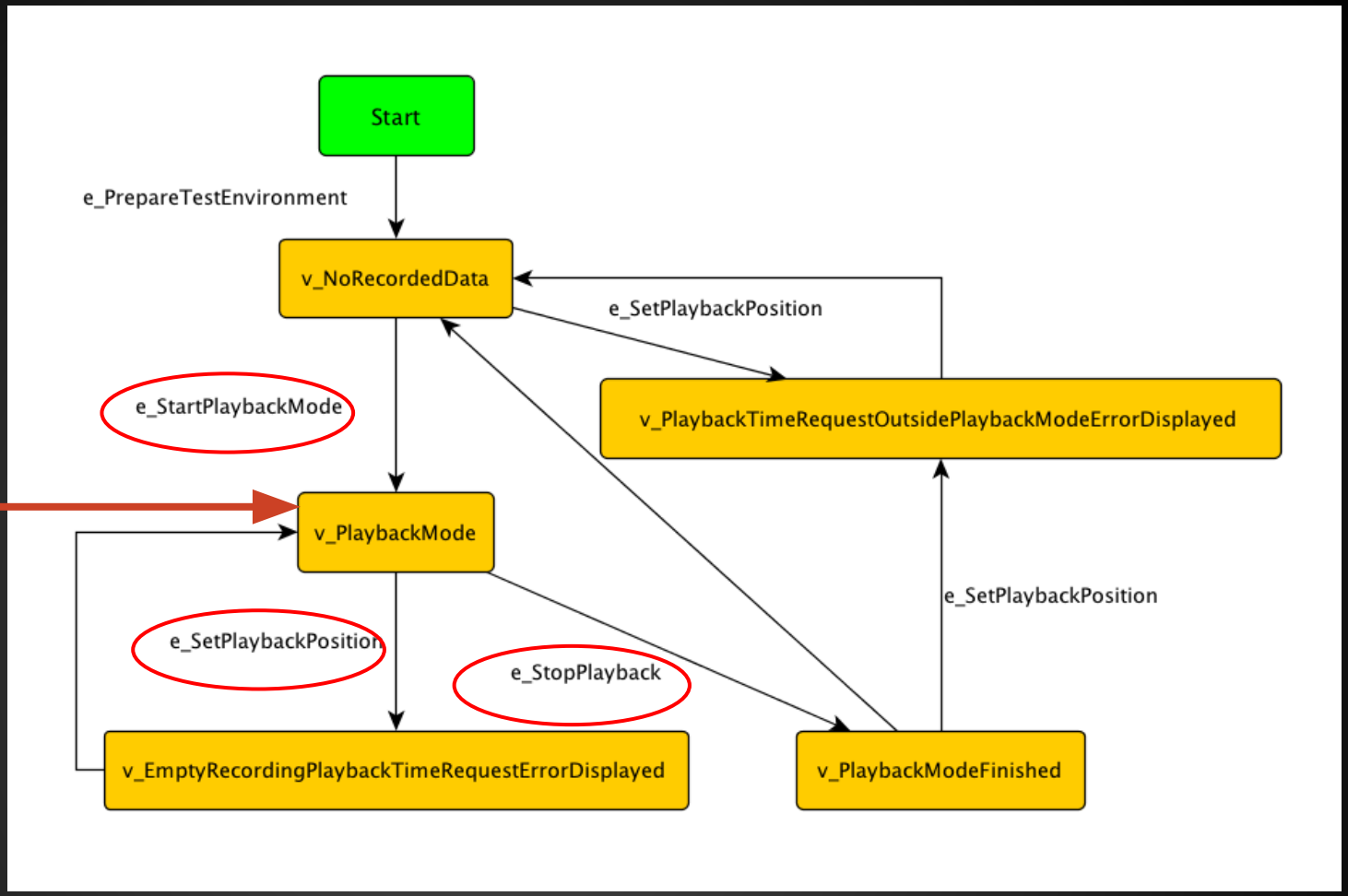


Animation State Machine Model

- Scenario: creating a simple state machine for animation
- Model created by non-programmers
- All logic covered by Action annotations
- Implementation required only 30 lines of code

unity

undesired system state



Animation Recording Model (negative test)

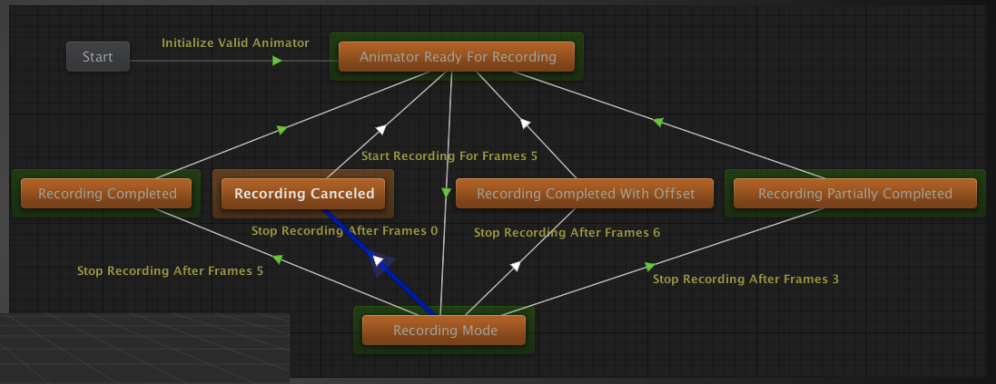
- Scenario: testing how animation playback system handles interaction with an empty animation recording
- Model design process uncovers unnecessary system states which translate to unwanted workflow complexity

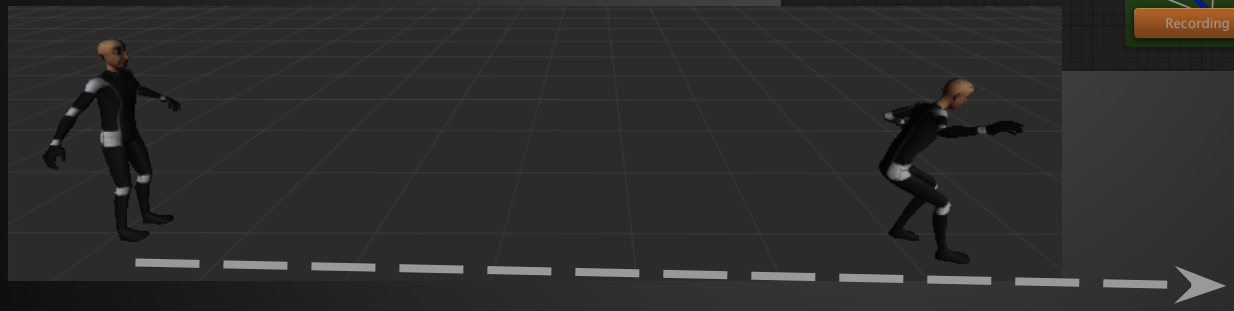
unity

Unity GraphWalker

Basic feature set

- Online and offline model traversals, coverage tracking
- Real-time and frame-by-frame preview and feedback
- Direct (double-click) access from model UI to code





Implementation

- Java GraphWalker compatibility
- Mono runtime / CaaS
- Coroutine-based execution

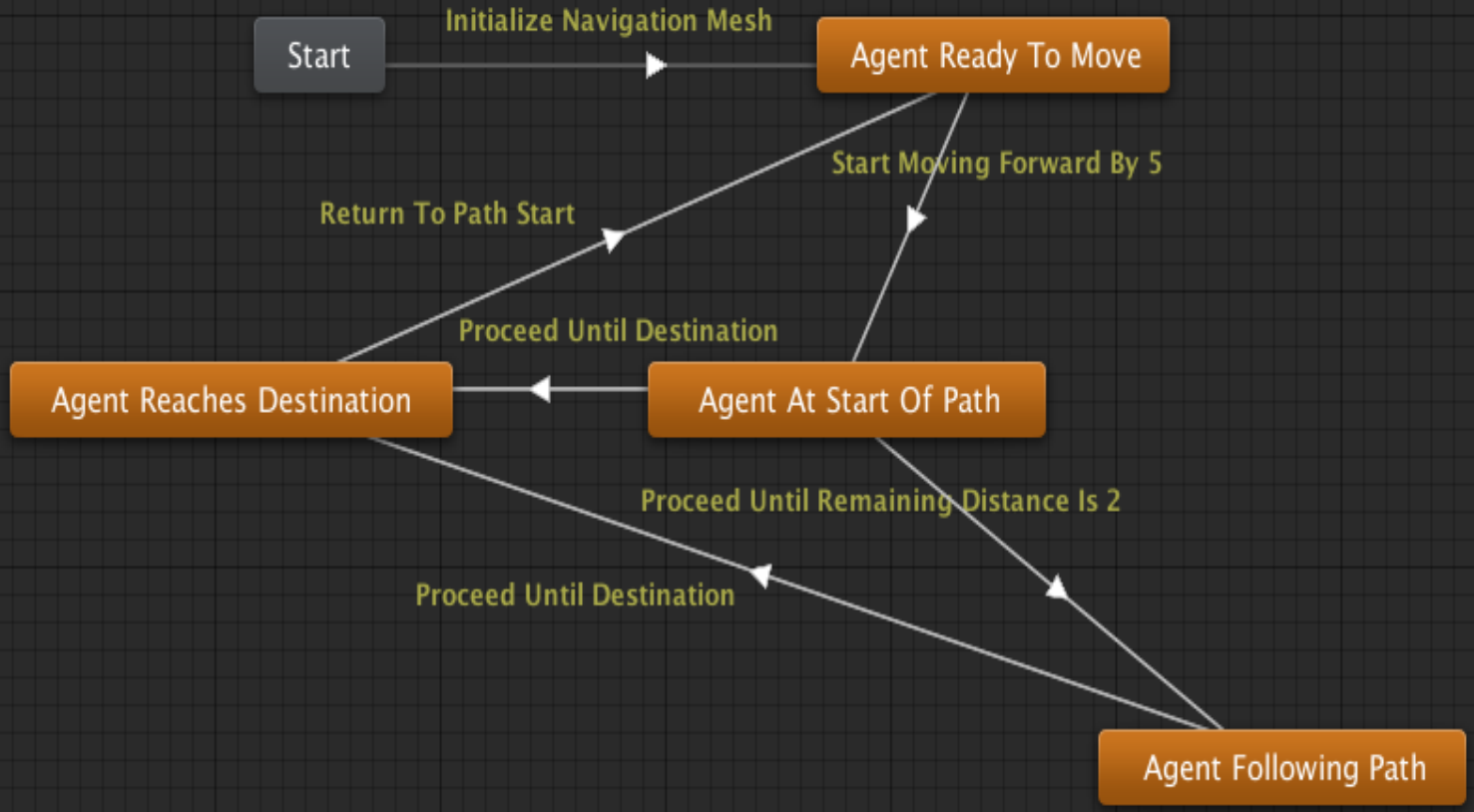
unity

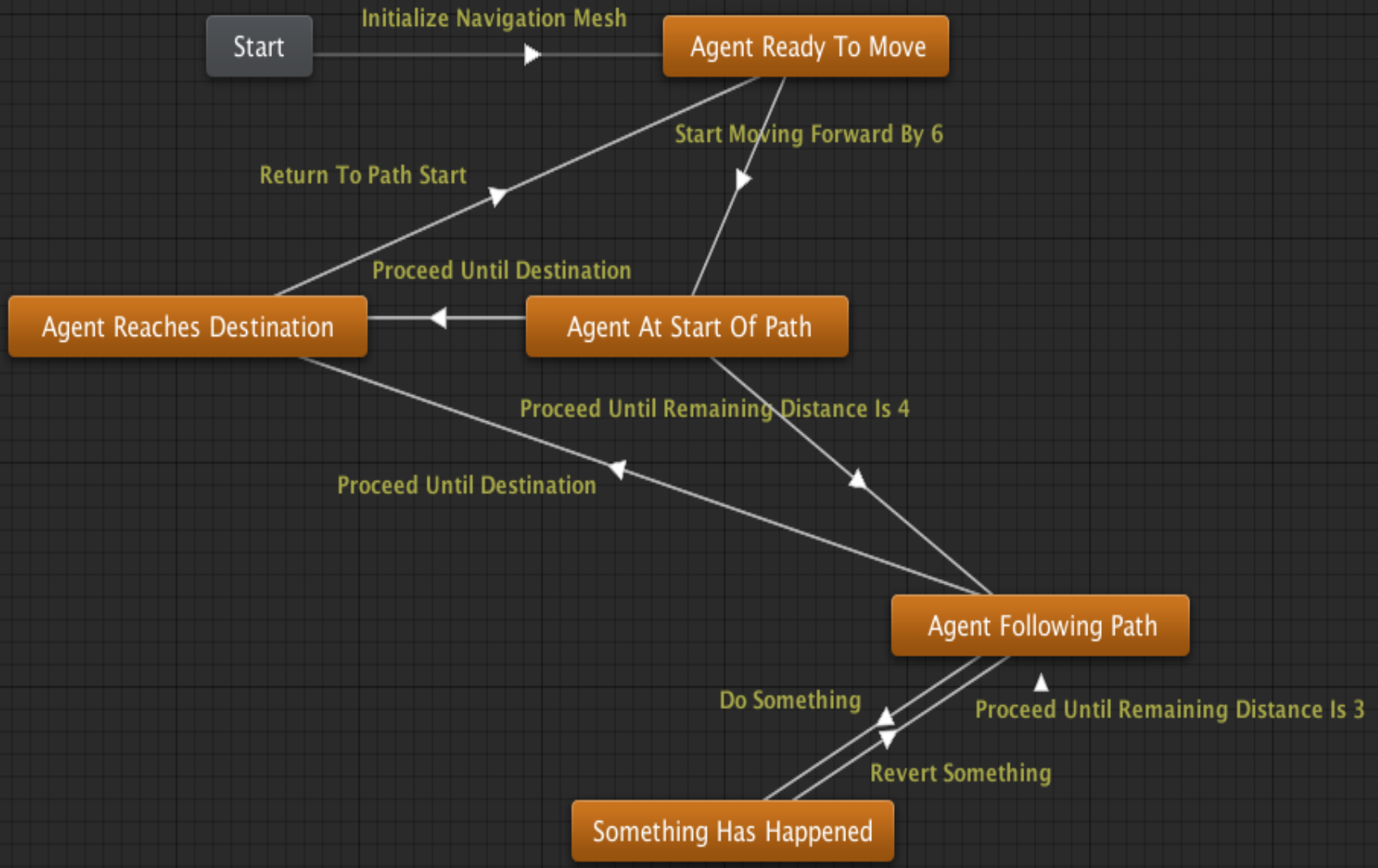
Navigation Mesh Pathfinding

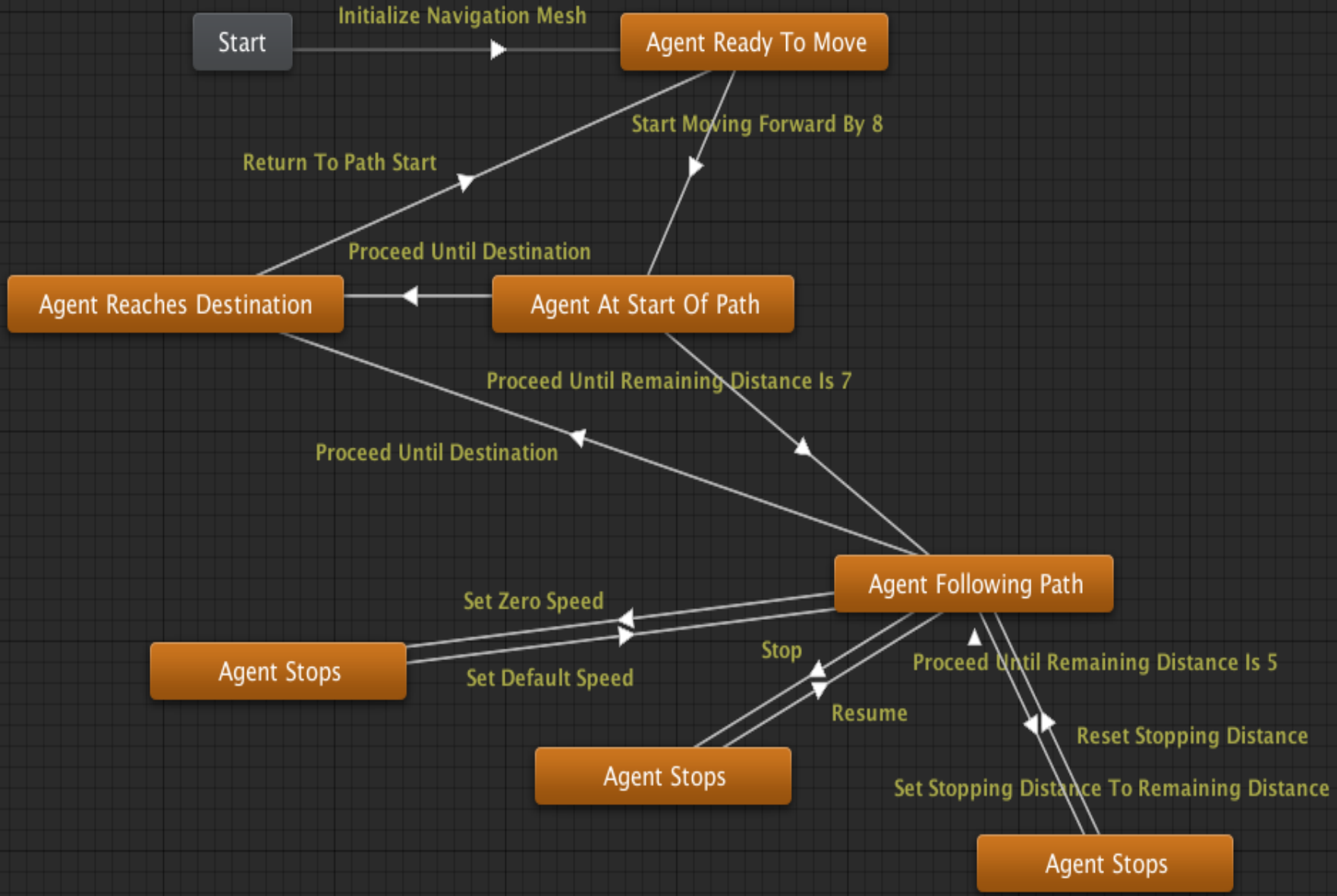
Navigation Mesh Pathfinding Model

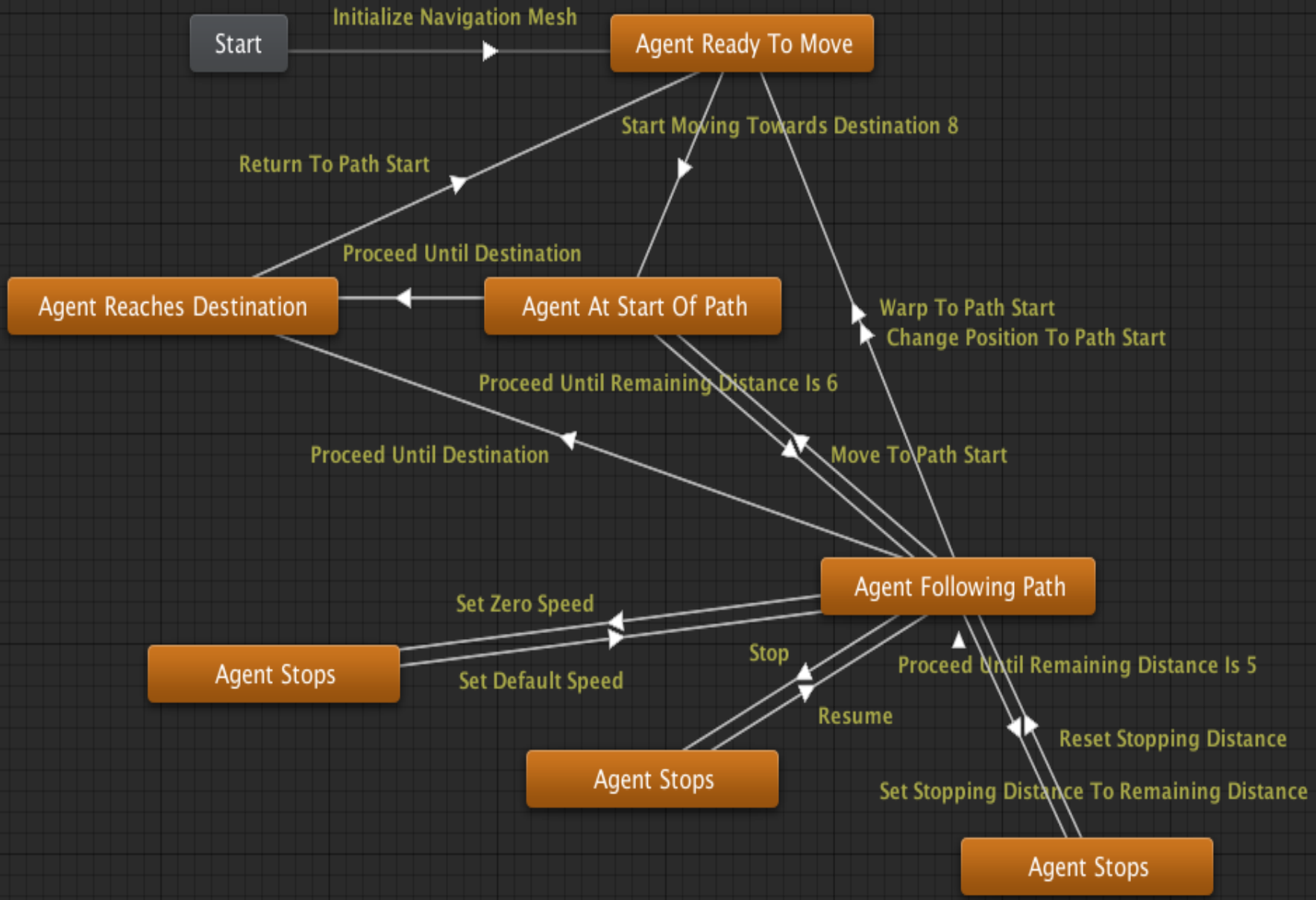
- Scenario: Path traversal towards a defined destination point
- The model scales rather quickly towards coverage equivalent to that of ~30 typical unit-level test scenarios
- Modular nature of the implementation makes it well suitable for integration testing against other sub-engines

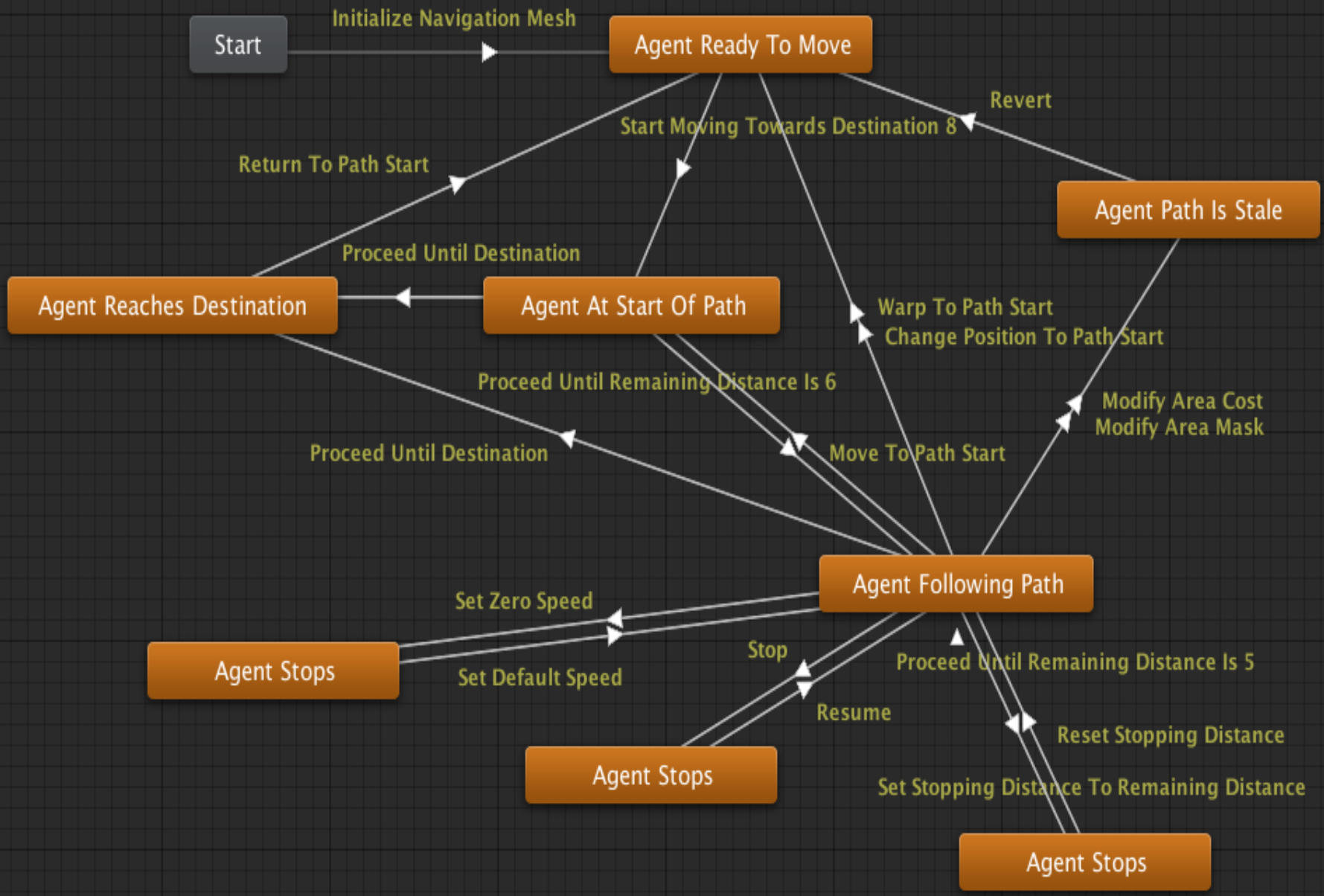
unity


Start
Initialize Navigation Mesh
Agent Ready To Move
Start Moving Forward By 5
Return To Path Start
Proceed Until Destination
Agent Reaches Destination
Agent At Start Of Path
Proceed Until Remaining Distance Is 2
Proceed Until Destination
Agent Following Path


Start
Initialize Navigation Mesh
Agent Ready To Move
Start Moving Forward By 6
Return To Path Start
Proceed Until Destination
Agent Reaches Destination
Agent At Start Of Path
Proceed Until Remaining Distance Is 4
Proceed Until Destination
Agent Following Path
Do Something
Proceed Until Remaining Distance Is 3
Revert Something
Something Has Happened


Start
Initialize Navigation Mesh
Agent Ready To Move
Start Moving Forward By 8
Return To Path Start
Proceed Until Destination
Agent Reaches Destination
Agent At Start Of Path
Proceed Until Remaining Distance Is 7
Proceed Until Destination
Agent Following Path
Set Zero Speed
Agent Stops
Set Default Speed
Stop
Proceed Until Remaining Distance Is 5
Resume
Reset Stopping Distance
Agent Stops
Set Stopping Distance To Remaining Distance
Agent Stops


Start
Initialize Navigation Mesh
Agent Ready To Move
Start Moving Towards Destination 8
Return To Path Start
Proceed Until Destination
Agent Reaches Destination
Agent At Start Of Path
Warp To Path Start
Change Position To Path Start
Proceed Until Remaining Distance Is 6
Proceed Until Destination
Move To Path Start
Agent Following Path
Set Zero Speed
Agent Stops
Set Default Speed
Stop
Proceed Until Remaining Distance Is 5
Resume
Reset Stopping Distance
Agent Stops
Set Stopping Distance To Remaining Distance
Agent Stops


Start
Initialize Navigation Mesh
Agent Ready To Move
Revert
Start Moving Towards Destination 8
Return To Path Start
Agent Path Is Stale
Proceed Until Destination
Agent Reaches Destination
Agent At Start Of Path
Warp To Path Start
Change Position To Path Start
Proceed Until Remaining Distance Is 6
Modify Area Cost
Modify Area Mask
Proceed Until Destination
Move To Path Start
Agent Following Path
Set Zero Speed
Agent Stops
Stop
Proceed Until Remaining Distance Is 5
Set Default Speed
Resume
Reset Stopping Distance
Agent Stops
Set Stopping Distance To Remaining Distance
Agent Stops

Conclusions

- **Findings**
  - Model Based Testing techniques are very well suited for structured workflow and scenario testing
  - Visual model design is a promising platform for sharing and maintaining test design ideas
  - Using a lightweight test design workflow often naturally encourages additional system exploration and leads to more interesting test scenarios

- **Demo**
  - http://files.unity3d.com/marek/mbt_demo.zip

- **Q & A**

unity