Live Game Design
RAAK-MKB project

CWI Scientific Meeting – March 31st 2017
Riemer van Rozen
CWI SWAT group

Joint work with:
Tijs van der Storm, Paul Klint
Problem Statement

• Problem
  – representation gap of game design: “the gap between a game’s design and its source code”

• Long game design iterations
  – prevent quickly experimenting with alternative game designs
  – game quality under pressure

• Missing
  – powerful notations for modifying a game’s elements
  – timely feedback
Objectives

• **Question**
  – Can the representation gap of game design be bridged with tools for exploring the design space?

• **How can tools help**
  1. shorten game design iterations and speed-up the design process
  2. closely match design to expertise and imagination
  3. help to improve the quality
  4. enable to design in a more targeted way
Approach: Live Game Design

- **Approach**
  - Live Intelligent Visual Environments for Game Design (Live Game Design)

- **Visual Programming Languages**
  - Visual notations for describing and steering interactive game elements (prototyping, fine-tuning) attuned to the expertise of game designers

- **Live feedback and feed-forward**
  - Immediate and continuous feedback on modification results
  - Design alternatives that can be inspected and applied to focus the creative design process
Live Textual Domain-Specific Languages

- **Domain-Specific Language (DSL) for the Game Domain:**
  Micro-Machinations is a language and library that enables game designers to modify a game’s rules at run-time.

- **Example:** Johnny Jetstream

**Source:**
```
kill income: kill -10 -> gold
pool gold is "$" at 20
cost: gold -10 -> buyHp
user converter buyHp
benefit: buyHp -20 -> hp
pool hp is "+" at 100
damage: hp -10 -> hit
drain hit
```

**Step 1:** Play Test v1

**Step 2:** Re-design

**Step 3:** Play Test v2
Machinations Evolution & Approach

2009: Conceptual Game Design Aid
2013: Formal Analysis + text, modules
2014: Live Adaptations
2015: A Pattern-Based Game Mechanics Design Assistant

Machinations

auto source s
pool p at 7
flow: s -p-> p

MM Lib

Gameplay Engineer  Player

Apply Design Alternatives

Mechanics Patterns

Mechanics Pattern Language

Mechanics Design Assistant

Game Mechanics
Advanced Game Design

Ernest Adams
Joris Dormans
Machinations Evolution & Approach


Live State Machine Language in Rascal

```
1 machine doors
2   state closed
3     open => opened
4   state opened
5     close => closed
6
7 end
```

```
<table>
<thead>
<tr>
<th>State</th>
<th>#</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed</td>
<td>1</td>
<td>[open]</td>
</tr>
<tr>
<td>opened</td>
<td>0</td>
<td>[close]</td>
</tr>
</tbody>
</table>
```
Current and Future Work

• **Question**
  – Can the representation gap of game design be bridged with tools for exploring the design space?

• **Approach**
  – Live Intelligent Visual Environments for Game Design (Live Game Design)

• **Work in progress**
  – Live Game Design project [http://livegamedesign.github.io/](http://livegamedesign.github.io/)
  – Generic frameworks for Live DSLs

• **Liked the live programming demo?**
  – Twitter @rvrozen

**References**

