Topic: HTML5-based Web Game Testing

Yibin Liao, Guodong Zhu
Web Game – Browser Game

Flash game
• 1997
• Install flash player plugin
• Not available on mobile browser

Html5 game
• 2011
• No plugin required
• Portable & cross-platform
Web Game – Browser Game

Flash game

Adobe is all about enabling designers and developers to create the most expressive content possible, regardless of platform or technology. For more than a decade, Flash has enabled the richest content to be created and deployed on the web by reaching beyond what browsers could do. It has repeatedly served as a blueprint for standardizing new technologies in HTML. Over the past two years, we’ve delivered Flash Player for mobile browsers and brought the full expressiveness of the web to many mobile devices.

However, HTML5 is now universally supported on major mobile devices, in some cases exclusively. This makes HTML5 the best solution for creating and deploying content in the browser across mobile platforms. We are excited about this, and will continue our work with key players in the HTML5 community, including Google, Apple, Microsoft and RIM, to drive HTML5 innovation they can use to advance their mobile browsers.

Our future work with Flash on mobile devices will be focused on enabling Flash developers to package native apps with Adobe AIR for all the major app stores. We will no longer continue to develop Flash Player in the browser to work with new mobile device configurations (chipset, browser, OS version, etc.) following the upcoming release of Flash Player 11.1 for Android and BlackBerry PlayBook. We will of course continue

Html5 game
Background

• HTML5-based Web game technologies
  • Html5
    • Canvas – a container for hosting graphics
  • JavaScript
    • Draw graphics on the Canvas
• WebGL
  • 3D graphic support
Drawing Graphics on the Canvas

Drawing on the Canvas with JavaScript

```html
<!DOCTYPE html>
<html>
<body>
  
  <canvas id="myCanvas" width="200" height="200"
    style="border:10px solid #2200c3;">
    Your browser does not support the canvas element.
  </canvas>

  <script>
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle = "#FF00CC";
    ctx.fillRect(0,0,150,75);
  </script>

</body>
</html>
```
HTML5 Game Example
HTML5 Game Example (source code)
HTML5 Game Example (source code)
What’s the problem?

• Current game testing technologies are mostly based on human effort.
What’s the problem?

• Manually play and play
  • Tedious Tasks
Goal

• Design an approach/framework for game testing (game path coverage, resource usage, detecting game errors, etc.) *without too much human effort (auto-play).*
Idea

• Construct a finite state machine to represent the game, then automatically play the game with the input generated based on the state path from the state machine for different test cases.
Finite State Machine

State Transition Table

<table>
<thead>
<tr>
<th>Input</th>
<th>State</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S₁</td>
<td>S₁, S₂</td>
</tr>
<tr>
<td>0</td>
<td>S₂</td>
<td>S₂, S₁</td>
</tr>
</tbody>
</table>

State Diagram

- S₁
- S₂
- Transition arrows: 1 → 0 → 1
- Initial state: S₁
- Input: 1, 0
Game State example (Initial)
Game State example (Play)
Game State example (Pause)
Game State example (Dead)
Finite State Machine

- **Initial**
  - Click on “START”

- **Dead**
  - If any targets reach left side

- **Pause**
  - Key press = “P”
  - Key press = “P”

- **Play**
Finite State Machine (Sub-States in Play)

Initial

Dead

Pause

Click on “START”

Key press = “P”

If any targets reach left side

Key press = “P”

Click on target

Click off target

Fire \uparrow
Score \uparrow
Accuracy \uparrow/\rightarrow

Fire \uparrow
Score = 0
Accuracy = 0

Fire \uparrow
Score \rightarrow
Accuracy \downarrow

Score = 0
Accuracy = 0
Approach

• Steps:
  1. Analyze the program and construct a Control Flow Graph (CFG)
  2. Define the state keys from the CFG
     • Define which variables to be monitored
  3. Define the state transition table
     • Define the possible state transitions
  4. Construct a finite state machine according to the state keys and the CFG
  5. Generate inputs for different test cases from the finite state machine
     1. Inputs that cover a specific state path or game path
     2. Inputs that cause a violation against the state transition table
     3. Etc.
  6. Reproduce the test cases in the web browser
Approach Framework Architecture

Analysis Module

Game Source Code → JavaScript Analyzer → Control flow graph → State Key → State Machine

Testing Module

Input Generator

Test Cases

Inputs that can reach a certain state → Web Browser

Inputs that cause a violation → Auto-play

Pre-defined State Transition Table

<table>
<thead>
<tr>
<th>Input State</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>S2</td>
<td>S2</td>
<td>S1</td>
</tr>
</tbody>
</table>
Challenges

Analysis Module
• How to define the state keys and construct the finite state machine from the control flow graph
  • Manual?
  • Automatic?

Testing Module
• How to generate the inputs from the finite state machine
  • Manual?
  • Automatic?
Related work

• [Modeling the HTML DOM and Browser API in Static Analysis of JavaScript Web Applications](#)
• [An Analysis of the Dynamic Behavior of JavaScript Programs](#)
• [Jalangi: A Selective Record-Replay and Dynamic Analysis Framework for JavaScript](#)
Implementations

• Working on...
Evaluation Plan

• Game path coverage
• Number of errors detected
• False positive & True positive
• Etc.