CS354R

DR SARAH ABRAHAM

TOOLS PIPELINE

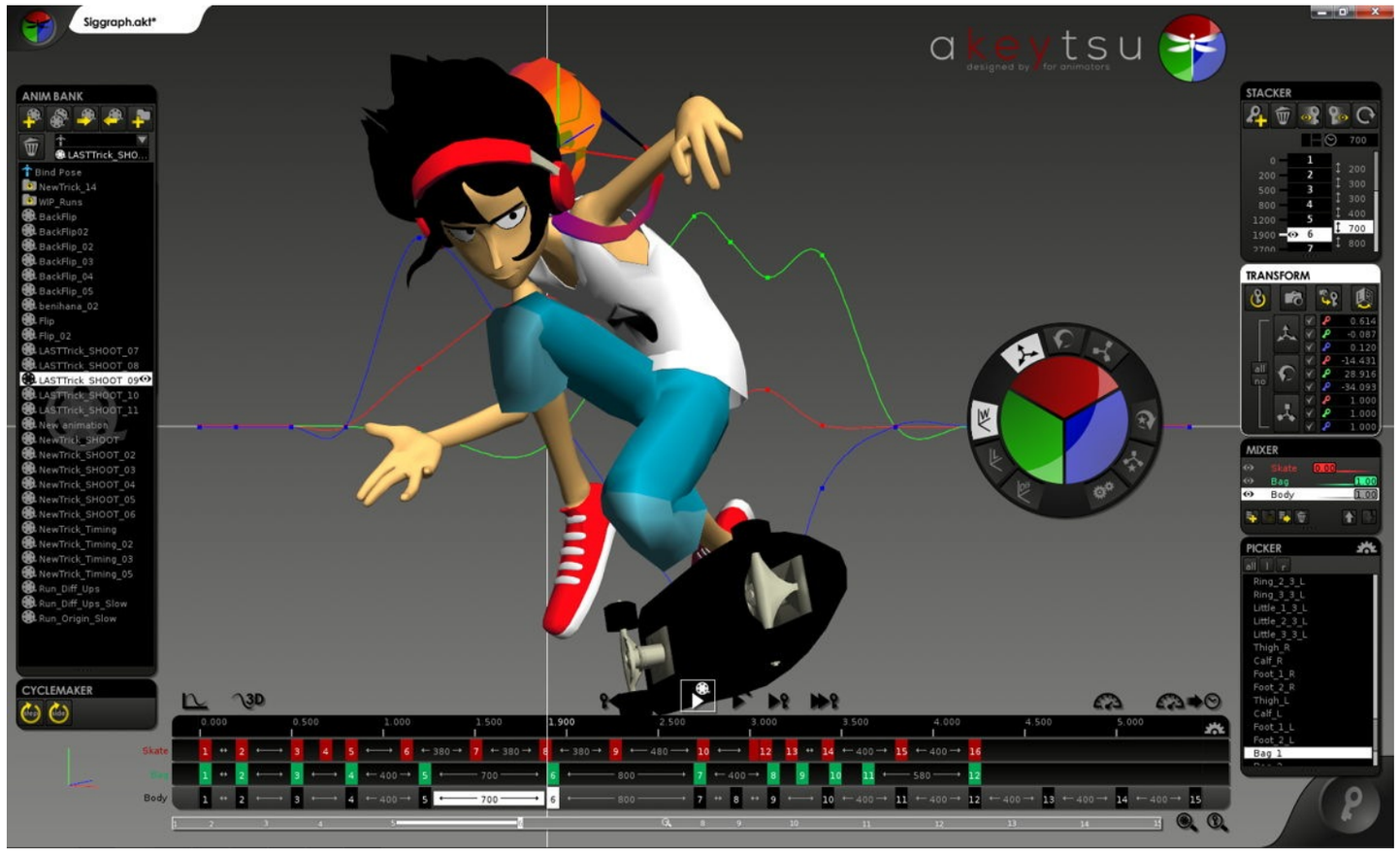
GAME AS MEDIA

- Games require art assets and design decisions
- Artists and designers may not have technical skills

(We can’t all be Daisuke Amaya)

"OFF THE SHELF" TOOLS FOR ARTISTS

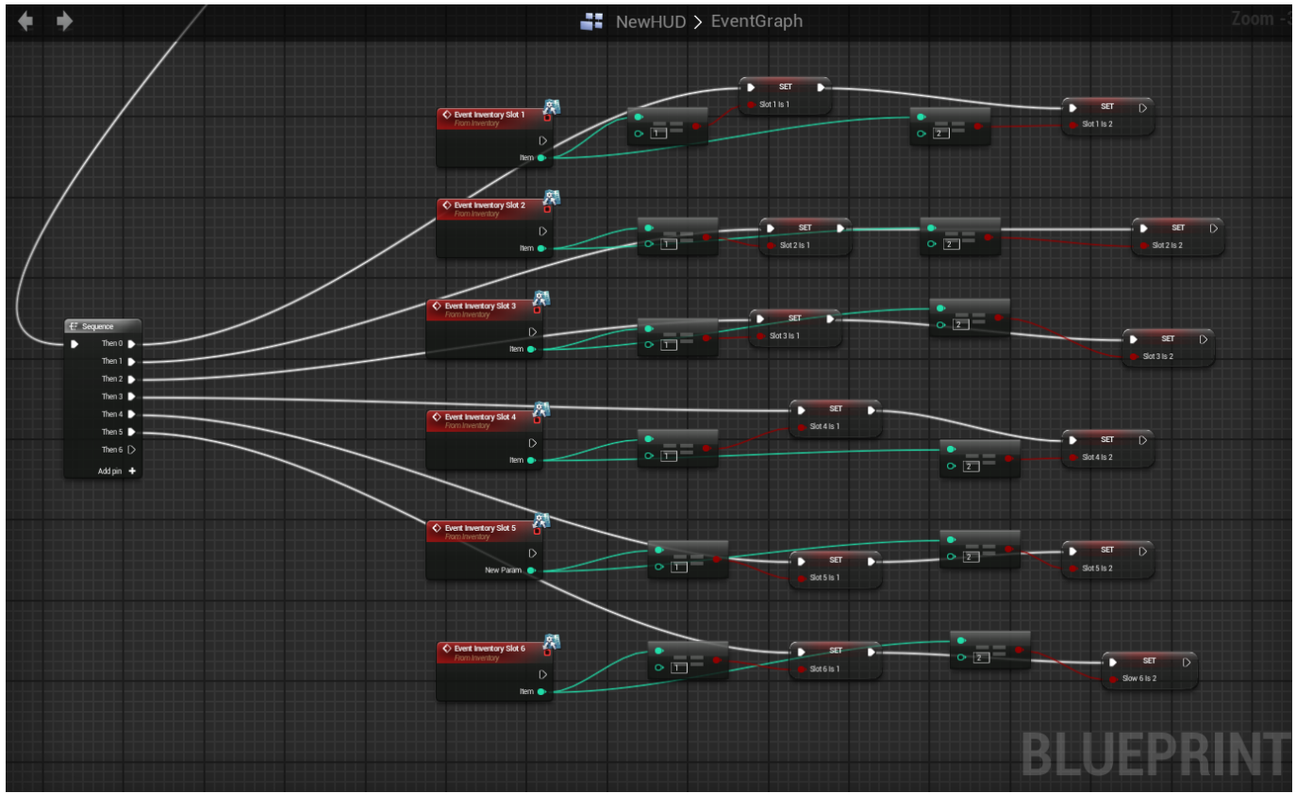
- Modeling
  - 3DS Max
  - Maya
  - ZBrush
- Texturing
  - Substance
  - Mudbox
  - Houdini
- Animation
  - Akeytsu
  - Maya



(Akeystu)

"OFF THE SHELF" TOOLS FOR DESIGNERS

- Level Editors
  - Tiled
  - In-Engine Level Editors
- Gameplay Editors
  - Blueprint
  - Excel
- UI Editors
  - Scaleform
  - Interface Builder
- Analytics
  - GamesAnalytics



(Blueprint)

IN-HOUSE TOOLS

- Create custom tools for game's art/design pipeline
- Modify existing tools for game's art/design pipeline
- These are inevitable parts of game development

DESIGNING IN-HOUSE TOOLS

- Understand the underlying data
- Listen to user suggestions
- Assume users have only intermediate expertise of tool

USER EXPERIENCE

- In tools development, the user is another developer rather than a player
- Consider:
  - What is the goal of the tool?
  - How will a user interact with the tool?
  - What does the user need to know?
  - What does the user not need to know?
  - What workflow will maximize productivity and **minimize unhappiness**?

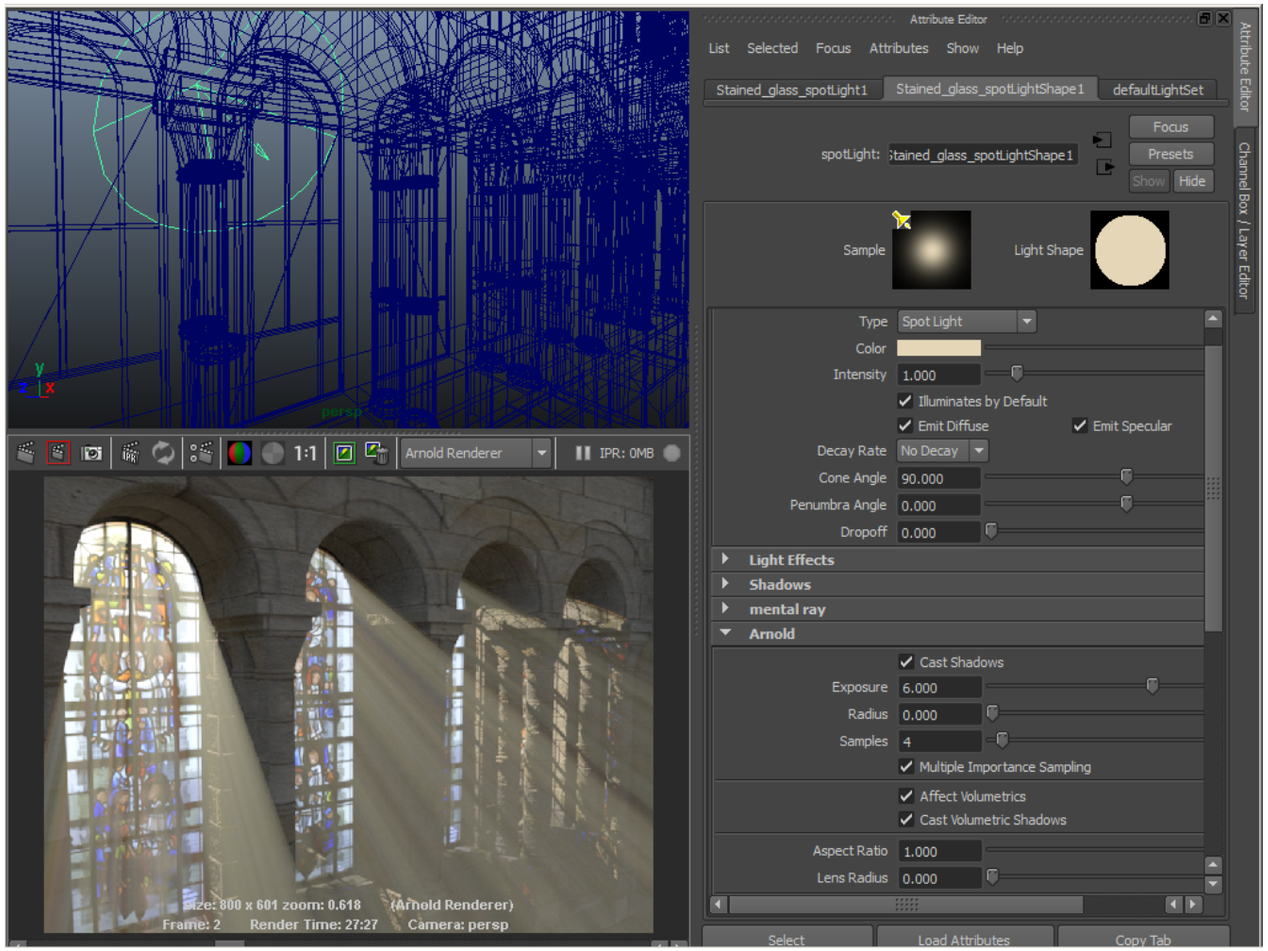
IN-CLASS ACTIVITY: DESIGNING TOOLS

- You have been asked to build the following tools:
  - A sequencer for AI in a puzzle game for scripting behaviors
  - A dialogue management system for branching dialogue
  - A combo editor for managing combat actions in a Musou-style game

- Consider the user experience of one of these
- Brainstorm some “first-pass” systems you might create to address designer needs

OTHER CONSIDERATIONS

- User does not need to understand all program details
- Tool should ideally have a short learning curve
- User should be protected
  - Limit inputs to valid data
- Abstract data as much as possible
  - Array of vertices are meshes
  - Lighting equations are parameterized sliders



(Maya)

EVALUATION: SYSTEM USABILITY SCALE (SUS)

- Survey-based
- Rate between 0 and 4
- 2.5 * Σ *ratings*
- 70-80 is average

The 10 statements adapted from the original SUS survey.

1. I think that I would like to use this tool frequently
2. I found the tool unnecessarily complex
3. I thought the tool was easy to use
4. I think that I would need the support of a technical person to be able to use this tool
5. I found the various functions in this tool were well integrated
6. I thought there was too much inconsistency in this tool
7. I would imagine that most people would learn to use this tool very quickly
8. I found the tool very cumbersome to use
9. I felt very confident using the tool
10. I needed to learn a lot of things before I could get going with this tool

BUILDING IN-HOUSE TOOLS

- Tools are glue between multiple systems
- …which is basically what you've been doing this entire course

- Which APIs are accessible?
- What formats can the systems read?
- How much interaction does your tool require?
  - Automated scripts
  - WYSIWYG interface

FILE INPUT/OUTPUT

- Simple, easy to read formats are ideal
  - XML
  - JSON
- Build parser with project requirements in mind
  - Over-engineering wastes time
  - Slap-dash code can become legacy
- Familiarity with the company's operating systems and libraries is probably necessary
  - Makefiles, library linking issues and all that good stuff

SCRIPTING

- You may have a choice
  - Python
  - Lua
  - Bash
  - Perl
- Or you may have to use something internal
  - MEL script
  - ActionScript
  - Your own?

WYSIWYG INTERFACES

- What You See Is What You Get
  - Well understood concept
  - Easy for less technical people to use
- Same rules for outward-facing GUIs apply to internal GUIs
- Clean, robust, intuitive interfaces lead to greater artist/designer productivity
  - Remember: asset creation is the most expensive, time-consuming part of the game development process!
- Note: good design and nice aesthetics are orthogonal issues
  - Which one is important in tool dev?

LEVEL EDITORS: IN-GAME VS. EXTERNAL

- In-game level editors
  - Seamless transition between designing and testing
  - May be harder to integrate libraries for GUI-creating tools
- External level editors
  - Many software solutions for the GUI interface
  - No direct connection between the creation tool and the game itself

SOME FINAL NOTES ON TOOL DESIGNS

- Incorporate hot keys for efficiency
- Ensure software stability
- Allow fast switches between “design” and “player” modes
- Don’t reinvent the wheel
  - Know what industry solutions already exist
  - Use those whenever most applicable

MODULES

- Very common concept in engine development
  - Use of smaller, independent libraries to bring in specific functionality
- Allows separation of “core” functionality from “specialized” functionality

GODOT MODULES

- Way of extending engine functionality in a portable way
  - Similar to “plugins” in other engines
- Modules placed in the `modules` subdirectory of the engine
  - Most Godot functionality already there -- including things like GDScript!
- More standard way of adding C++ functionality
  - We don’t use it because it requires modifying the actual engine source which isn’t practical on the lab machines

TYPICAL GODOT MODULES

- Bindings for an external library
- Optimizations for performance-critical parts of the game
- Adding new functionality to the engine or editor
- Porting an existing game
- You just love working in C++

CREATING A MODULE

- Create a folder within the `module` folder
  - Need a SCsub and config.py file for building
    - Can include any Godot headers/functionality required
- Will need to work in your own version of the engine
  - Lab machine access to the engine source not available
- Also, important to figure out *what* you are creating and *how* and *expected functionality* before you start coding...

THE FEW, THE PROUD . . .

- Tools writing is a dirty, thankless job
- But they make the world a better place!

REFERENCES

- Dan Goodman. Game Tools Tune-Up: Optimize Your Pipeline Through Usability <http://www.gamasutra.com/view/feature/132407/game_tools_tuneup_optimize_your_.php>

- Vijay Pemmaraju. Make Your Life Easier: Build a Level Editor <http://gamedevelopment.tutsplus.com/articles/make-your-life-easier-build-a-level-editor--gamedev-356>

- Richard Rouse III. Designing Design Tools <http://www.gamasutra.com/view/feature/131850/designing_design_tools.php>