Rensselaer

Submitty
Rensselaer Center for Open Source

Autograding Interactive Computer Graphics Assignments

Evan Maicus,
Matthew Peveler,
Andrew Aikens,
Barbara Cutler

SCAN ME

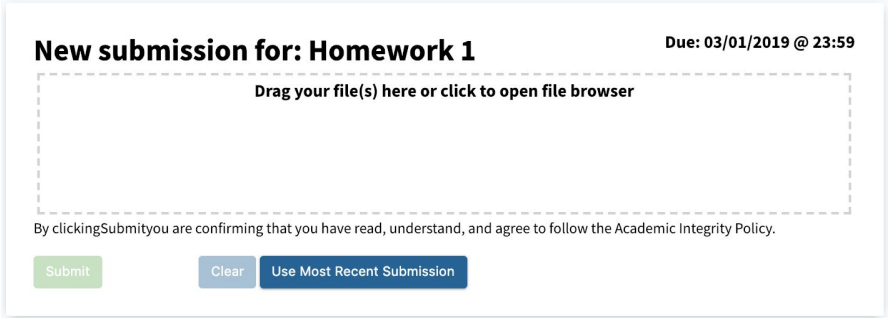
RCOS
Rensselaer Center for Open Source

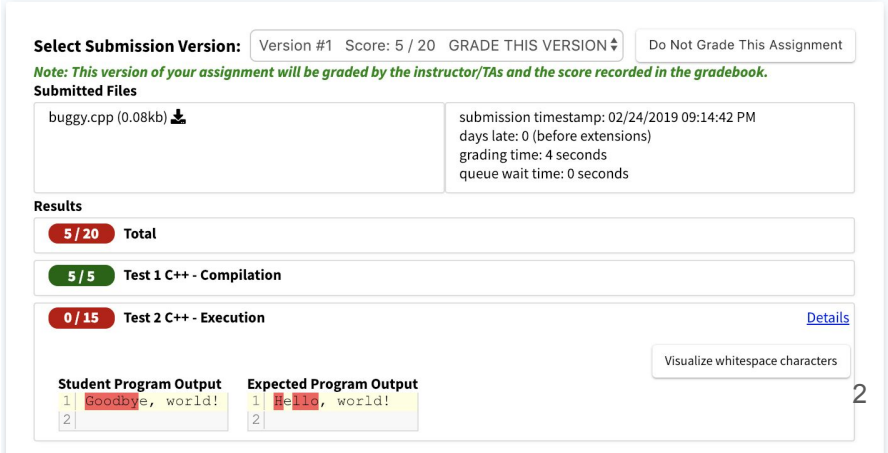
https://submitty.org/

Submitty

Rensselaer Center for Open Source

- A free, open source autograding platform.
  - ~2500 users
  - 12-15 courses supported per term at RPI
  - In operation since 2014
- Support for:
  - Assignment submission
  - Autograding
  - Exam grading/scanned PDF upload
  - Course communications (email/forum)
  - Course material hosting
  - Plagiarism detection
  - TA help queue

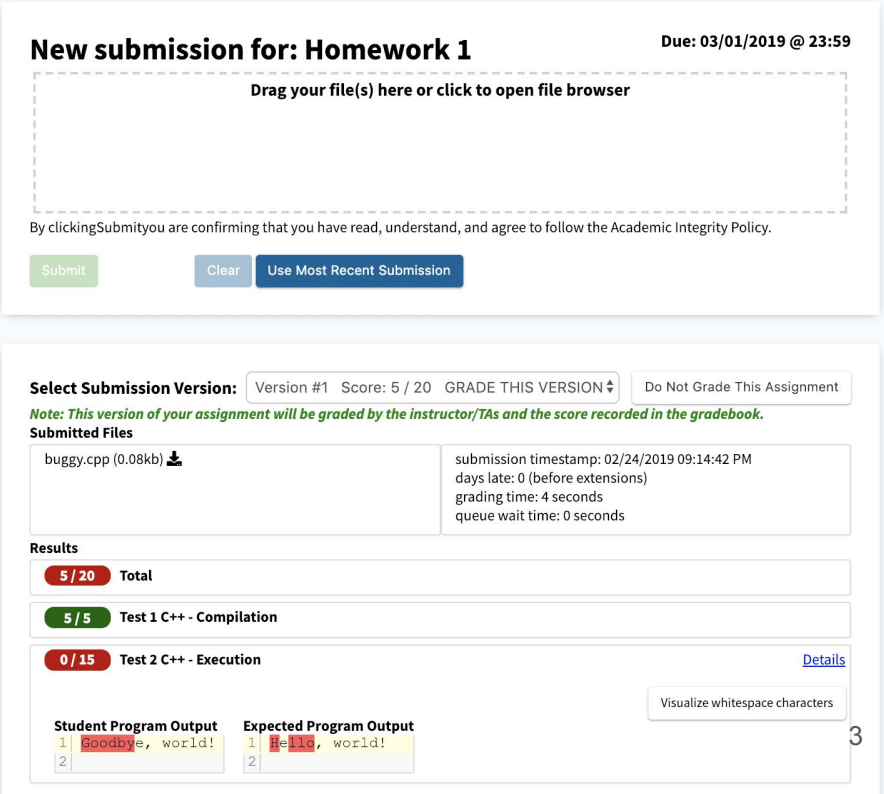




Autograding Configuration

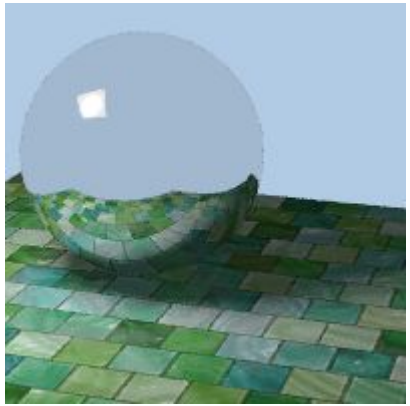
Submitty
Rensselaer Center for Open Source

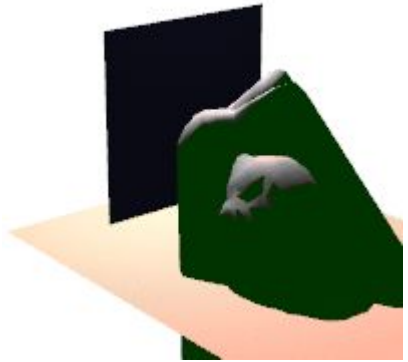
```
{
  "testcases" : [
    {
      "type" : "Compilation",
      "title" : "C++ - Compilation",
      "command" : "clang++ -Wall -o a.out -- *.cpp",
      "executable_name" : "a.out",
      "points" : 5
    },
    {
      "title" : "C++ - Execution",
      "command" : "./a.out input_file.txt",
      "points" : 15,
      "validation" : [
        {
          "method" : "diff",
          "actual_file" : "STDOUT.txt",
          "description" : "Program Output",
          "expected_file" : "test1_output.txt"
        }
      ]
    }
  ]
}
```

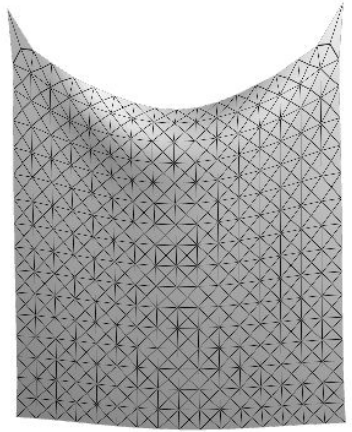


Motivation

- Course sizes are swelling [Wilcox, 2016]
- We need to autograde everything we can
- We aim to autograde graphical submissions







Scope

This work addresses:

1. Desktop applications
2. Display visual information on a screen (OpenGL)
3. Take user input (keyboard/mouse/stdin)
4. Often without a GUI
5. Applications with a unique, correct solution

This work is a step towards:

1. Grading web applications
2. Grading applications with many solutions

Challenges of Grading Graphics Assignments

Logistics:

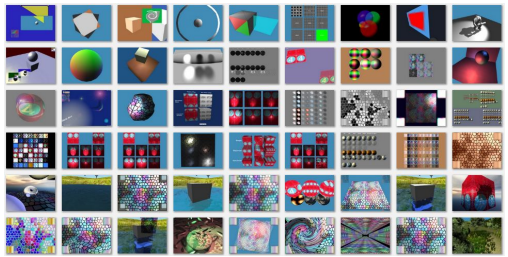
1. Longrunning (e.g. a raytracer)
2. Runs require attention
3. Differences in drivers/machines

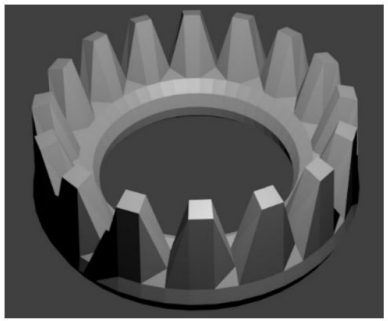
Determining Correctness:

1. Perceptual vs absolute correctness
2. Single vs a class of correct solutions
3. Partial Credit

Related Work

1. Perceptual vs. absolute correctness [Timofeitchik et. al. 2013]
2. Overcoming hardware differences [Pranckevičius, 2007, 2011]
3. Evaluate using geometric approximation [Heckbert et. al. 1997, Sahasrabudhe et. al. 2018,]



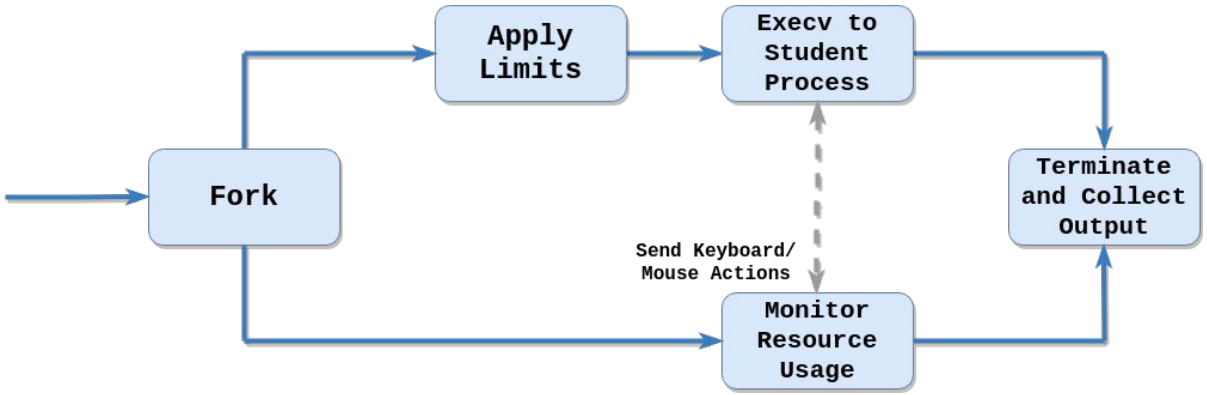


Building Our System: Requirements

- Securely invoke graphical applications
- Facilitate scripted interaction
- Capture graphical output
- Assist in automated and manual grading
- Provide students with actionable feedback

Building Our System: Invocation and Security

- Integration with Submitty's jailed sandbox
- Interact only with student windows
- Fault Tolerance -- a crash may occur at any time



Building Our System: Scripted Interaction

Input Actions:

Type, Stdin, Move Mouse, Click, Click and Drag

Output Actions:

Screenshot, GIF

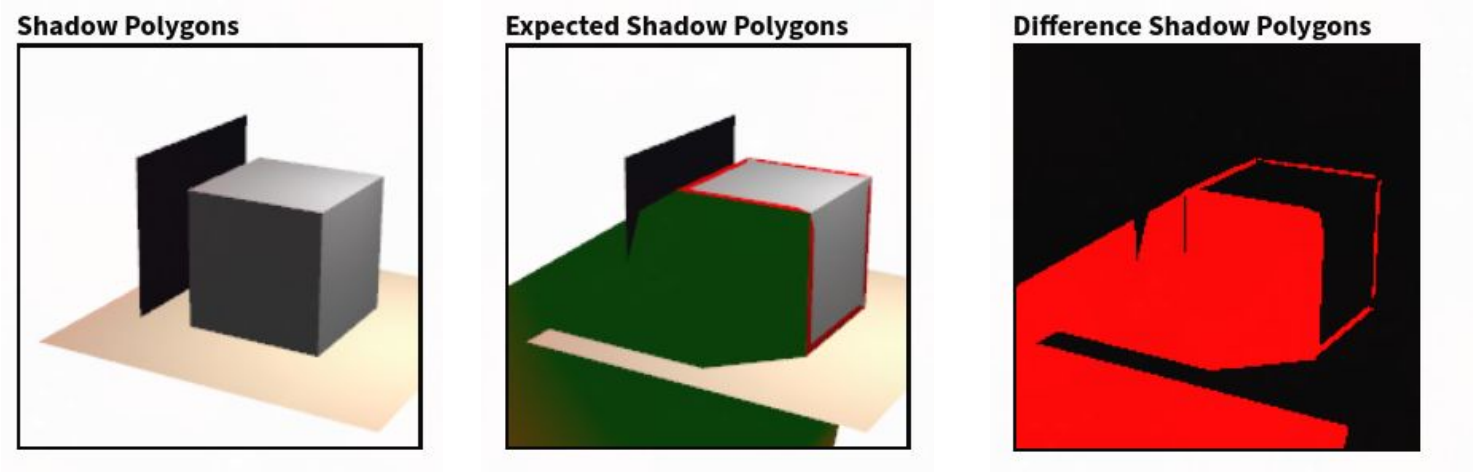
Sequence Actions:

Delay

```
[
  {
    "action" : "click and drag delta",
    "end_x" : 100
  },
  {
    "action" : "key",
    "key_combination" : "a"
  },
  {
    "action" : "gif",
    "seconds" : 2,
    "name" : "gif_1"
  },
  {
      "action" : "delay",
      "seconds" : 1
  },
  {
    "action" : "key",
    "key_combination" : "a"
  },
  {
      "action" : "screenshot",
      "name" : "screenshot_1"
  }
],
```

Building Our System: Automated Grading

1. **Utilize Provided Image Difference Graders**
   a. MSE Image (with support for SSIM)
   b. Configurable tolerance
2. **Compare against Instructor Output**
   a. Generated by our system given an instructor solution



Building Our System: Custom Validation

- Instructor written in Python
- Can import popular image comprehension packages
  - **numpy, scipy, scikit-image, opencv, Pillow, etc.**
- Input:
  - Student screenshots / GIFs / STDOUT
  - Instructor provided files
- Output
  - Score
  - Custom message
- Examples use cases:
  - Transform output into an easier-to-grade state
  - Perform classification algorithms on output
  - Evaluate image for accessibility (e.g. colorblindness)

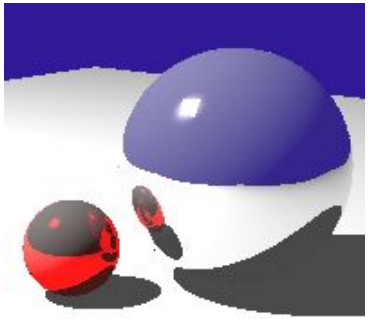
Case Study: Advanced Computer Graphics S19

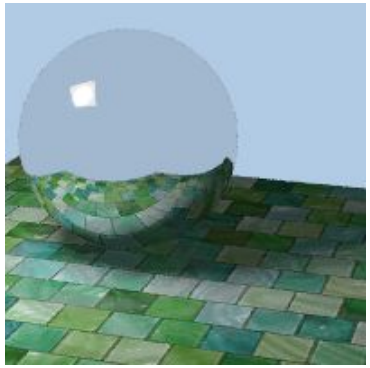
- Five OpenGL computer graphics assignments
  - Previously graded manually
- >30 students
- Variety of assignments
  - Longrunning
  - Simulation
  - Creative (open to interpretation)
- Deployment was successful
  - Many lessons learned

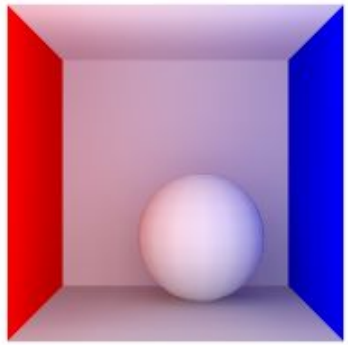
Grading Time-Intensive Computation

Ray tracing/radiosity/photon mapping assignment

- Testcases took >1 min
- 9 Testcases
- MSE error detection was particularly useful
- Automated tests were likely more patient than previous manual graders





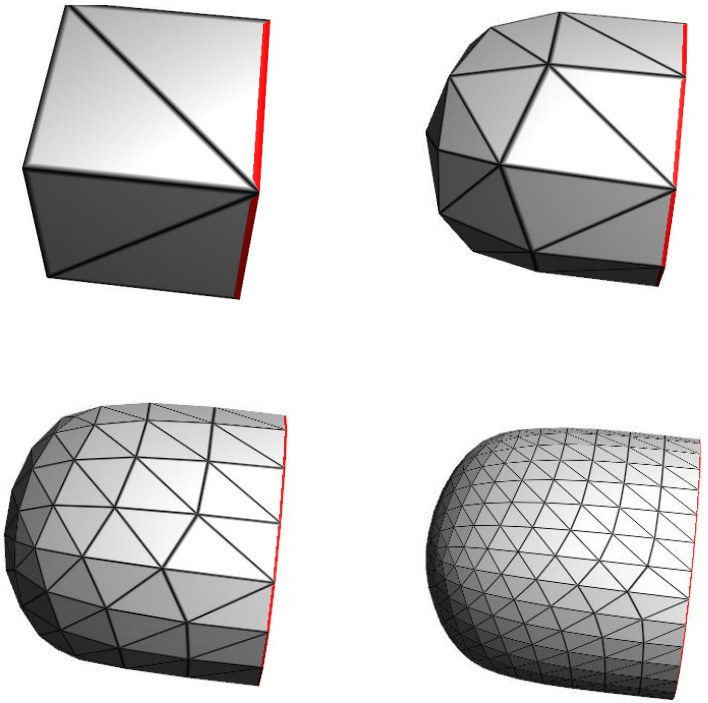


Step by Step Visual Unit Testing

- Decompose tests into intermediate screenshots

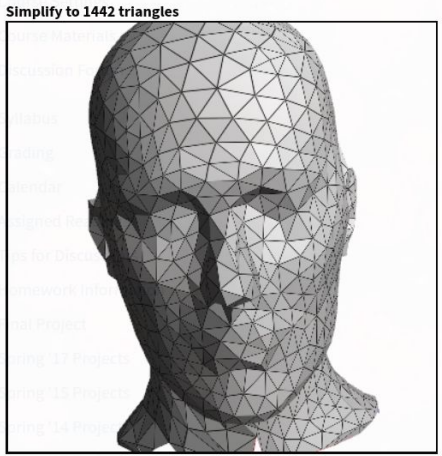
Useful For:

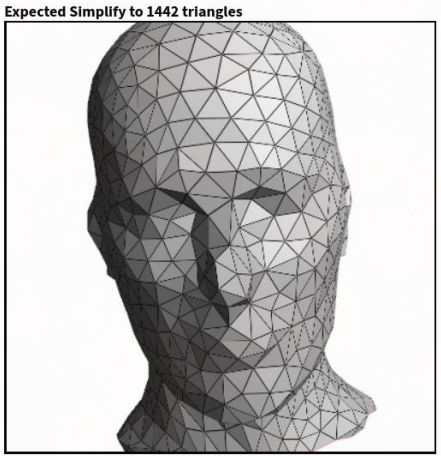
- Iterative Algorithms
- Dynamic scenes (lighting)
- Student debugging



Wireframe and Mesh Operations

- Add wireframe when grading mesh operations
- Allows finer detail to be captured
- Does not help students much with debugging





Physics Simulation Videos / Manual Grading

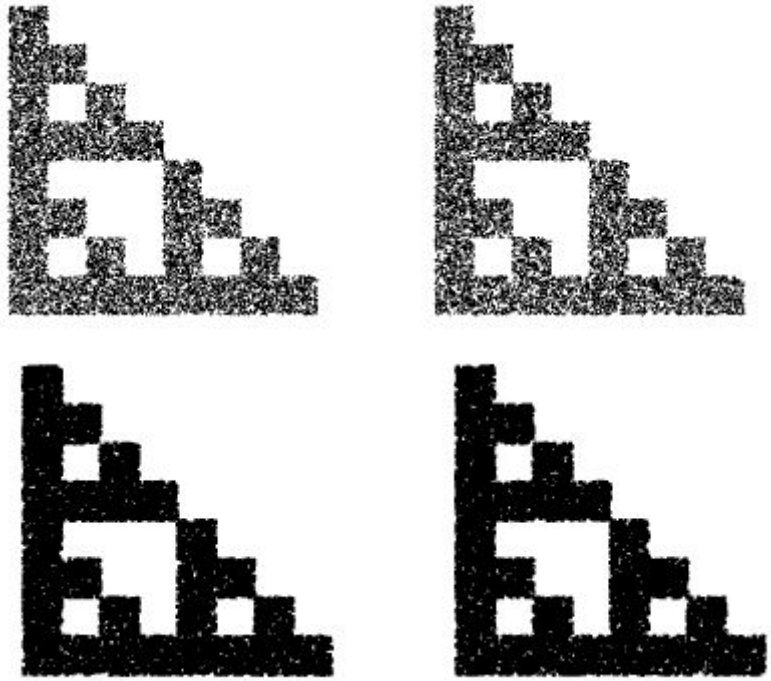
Simulations:

- Occur over multiple frames
- Difficult to tell from a single frame whether output is correct.
- GIFs + Manual Grading are helpful.

Creative Solutions for Nondeterminism

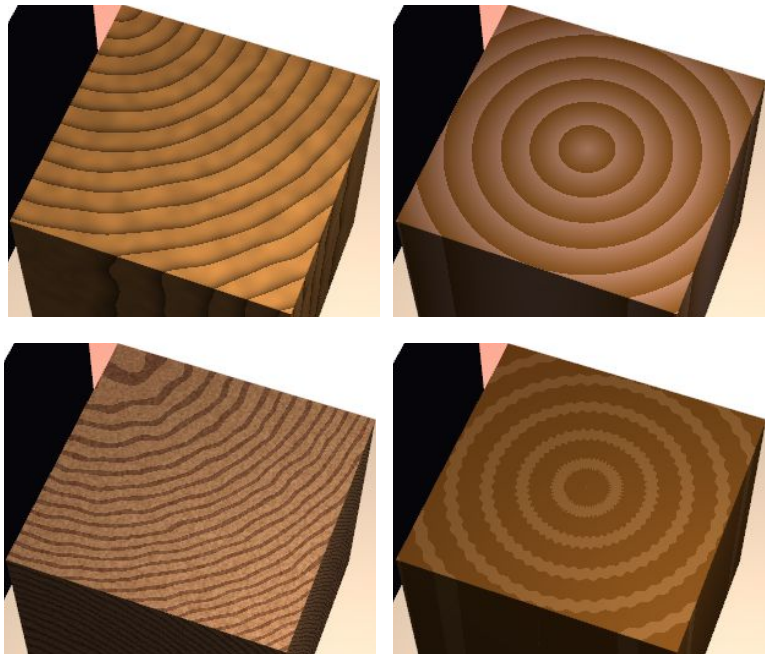
Fractal Assignment:

- Introduced students to transformations
- Naive Image Differencing Failed
- **Solution:** Increase point size



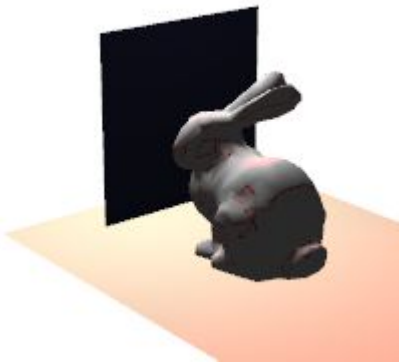
Automated Execution Without Validation

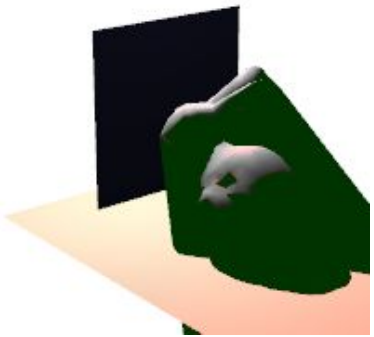
- Asked students to implement fragment/vertex shaders
- Instructions were left open-ended to promote creativity
- Results were gathered, then manually graded

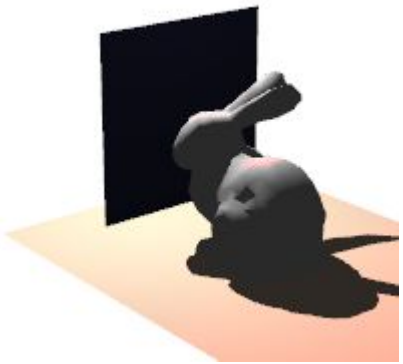


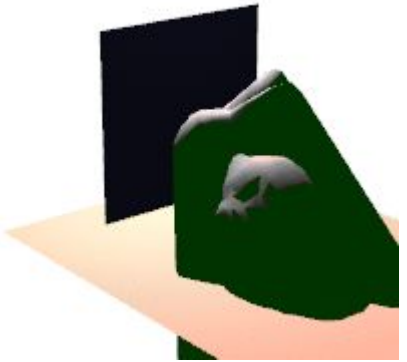
Generating Actionable Feedback

- Incremental Screenshots
- Video GIFs
- Difference Images
- STDOUT / Program Output





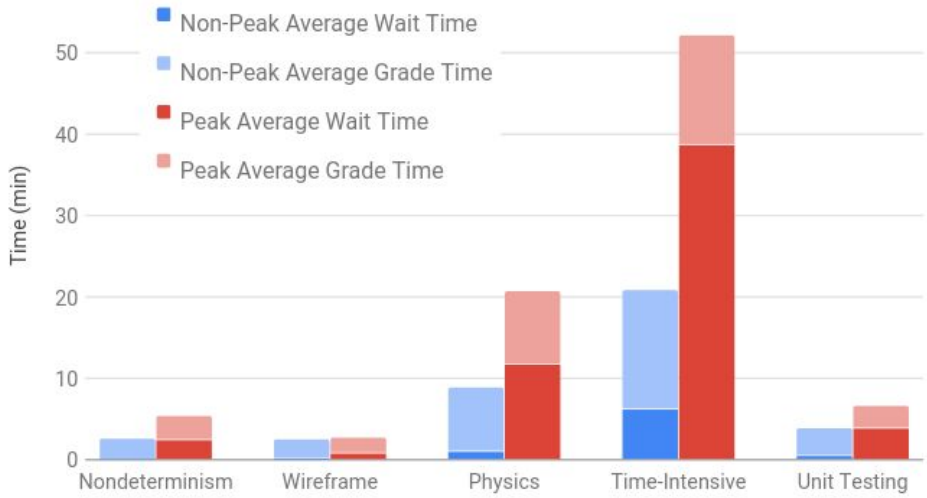




Post-Semester Study

Post-Semester Survey

- 27/34 students responded.
- 19 found our extension helpful, 6 did not answer, 2 found it unhelpful
- 4 commented on long wait times
- 18 felt the autograder didn’t offer enough partial credit, but many commented that TA intervention helped mitigate the issue
- Difference images and GIFs were viewed as helpful in debugging



Future Work

- **Autograding performance optimization and transparency**
  - Add testcase dependencies to allow early termination
  - Provide expected processing times to students
- **Ease of use utilities**
  - Directly record instructor input actions
- **More sophisticated built-in image comparison**
  - Add additional provided image comparison schemes

Contributions

Submitty
Rensselaer Center for Open Source

- **A system to run student graphics assignments**
  - Securely Executes Student Submissions
  - Built within an Open Source Autograding Platform
  - Captures Student Output
- **An interface to interact with student graphics submissions**
  - Provides Keyboard/Mouse Interactions
  - Captures Screenshots and GIFs
- **Provide actionable information to students**
  - Difference Images, Partial Executions, GIFs
- **Useful even when not used for autograding**

SCAN ME

Rensselaer

https://submitty.org/

Rensselaer

Submitty
Rensselaer Center for Open Source

Thank You

Evan Maicus,
Matthew Peveler,
Andrew Aikens,
Barbara Cutler

SCAN ME

RCOS
Rensselaer Center for Open Source

https://submitty.org/