A Flexible Simulation Framework for Graphics Architectures

Jeremy Sheaffer, David Luebke, Kevin Skadron

University of Virginia

Motivation

- No GPU simulators available in academia
  - Vendor simulators not available to academics
    - Probably lack necessary flexibility
- SimpleScalar has made a huge impact in the academic GP architecture community
  - Required a few years before work became interesting to industry
    - Started with incremental ideas that industry had already considered or been considering
  - Now 150-300 papers/year use SimpleScalar
    - 30%-50% are looked at by industry
    - At least 1%-2% of the ideas actually make it into products
- Hope to elicit the same kind of innovations in the GPU community

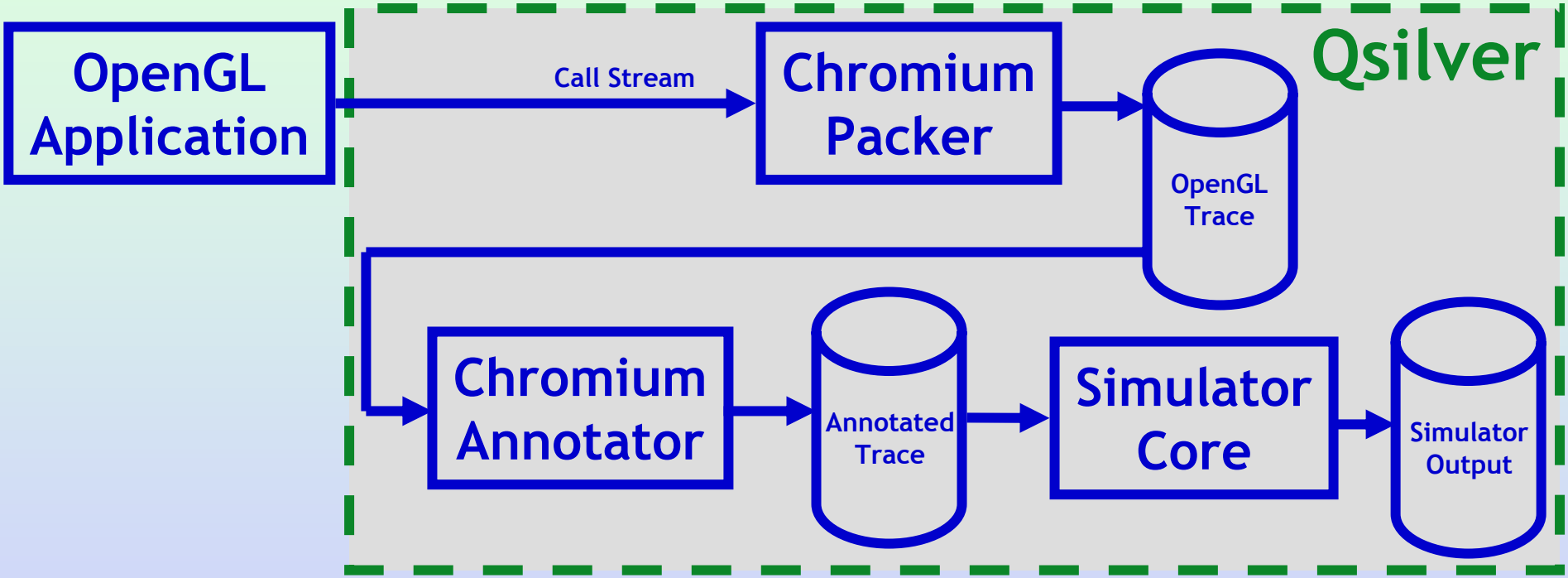
Qsilver

- An architectural simulator for GPUs
- Makes possible academic study of a wide array of architectural techniques
- Runtime configurable
- Traces any OpenGL application
- Small, extensible code-base

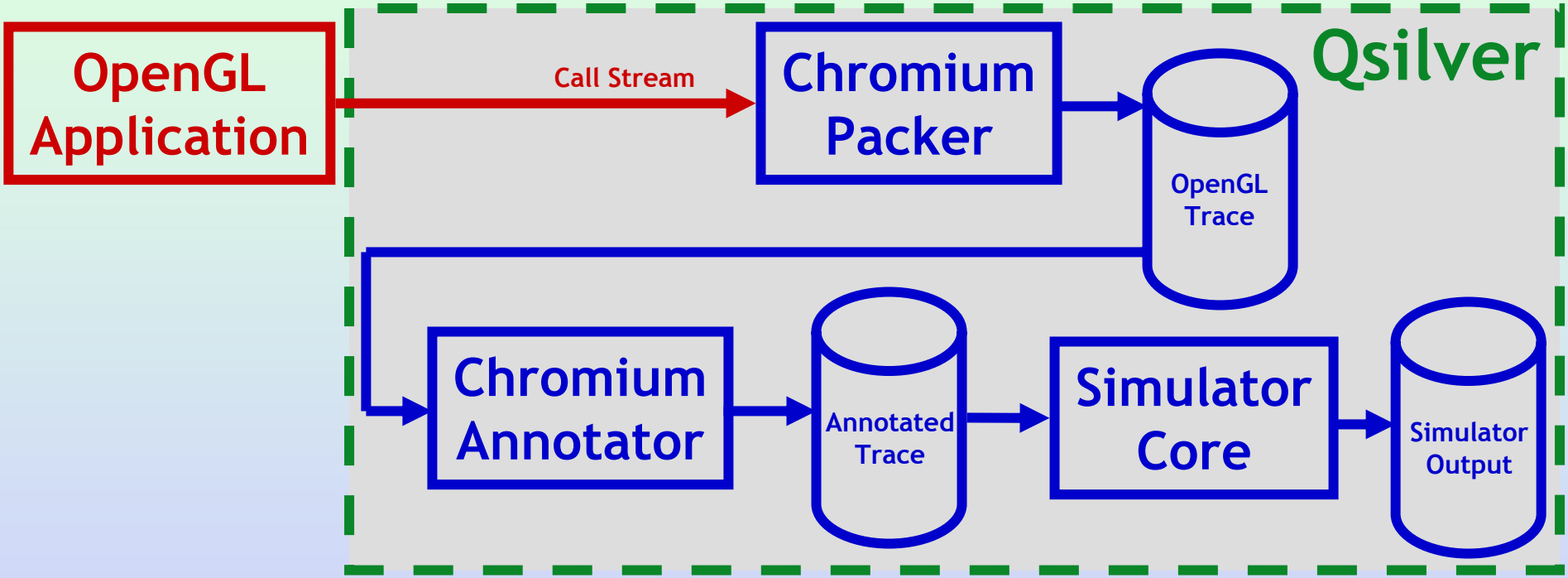
Chromium

- OpenGL stream interceptor and transformer
- Allows easy manipulation of the OpenGL call stream
- Usually used for parallel rendering applications
- No need for source code of OpenGL application

Application to Simulation

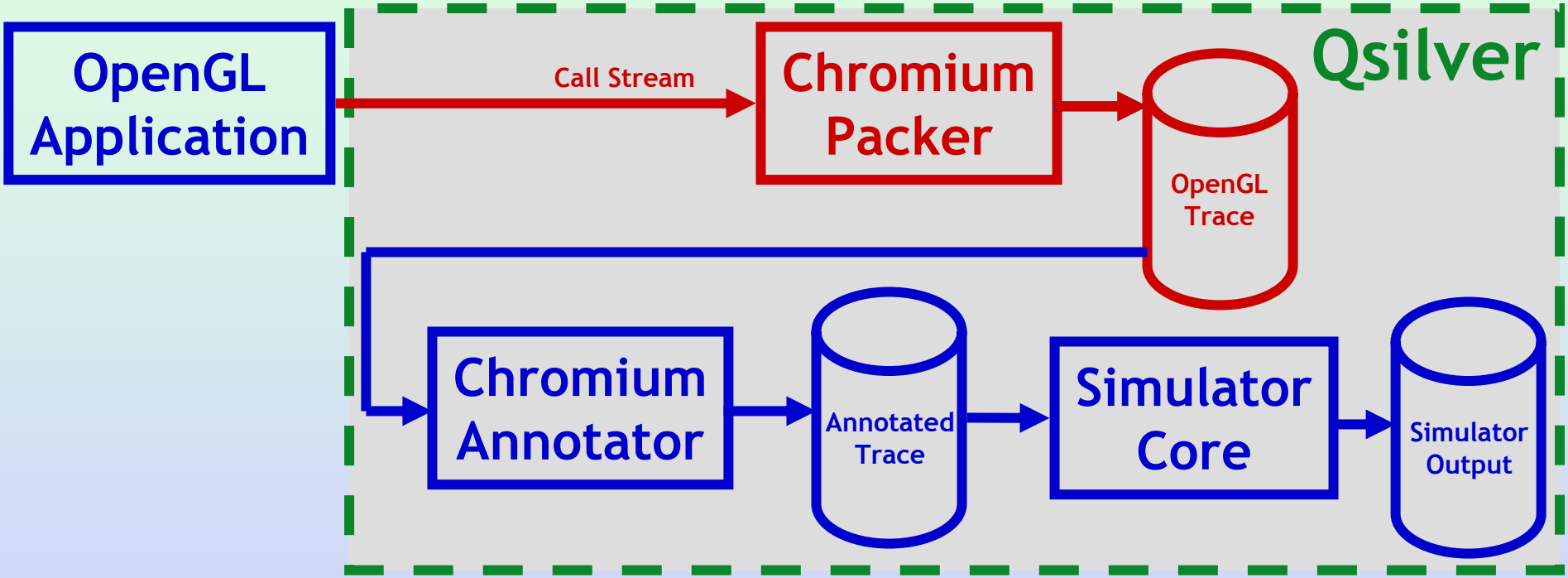


Application to Simulation



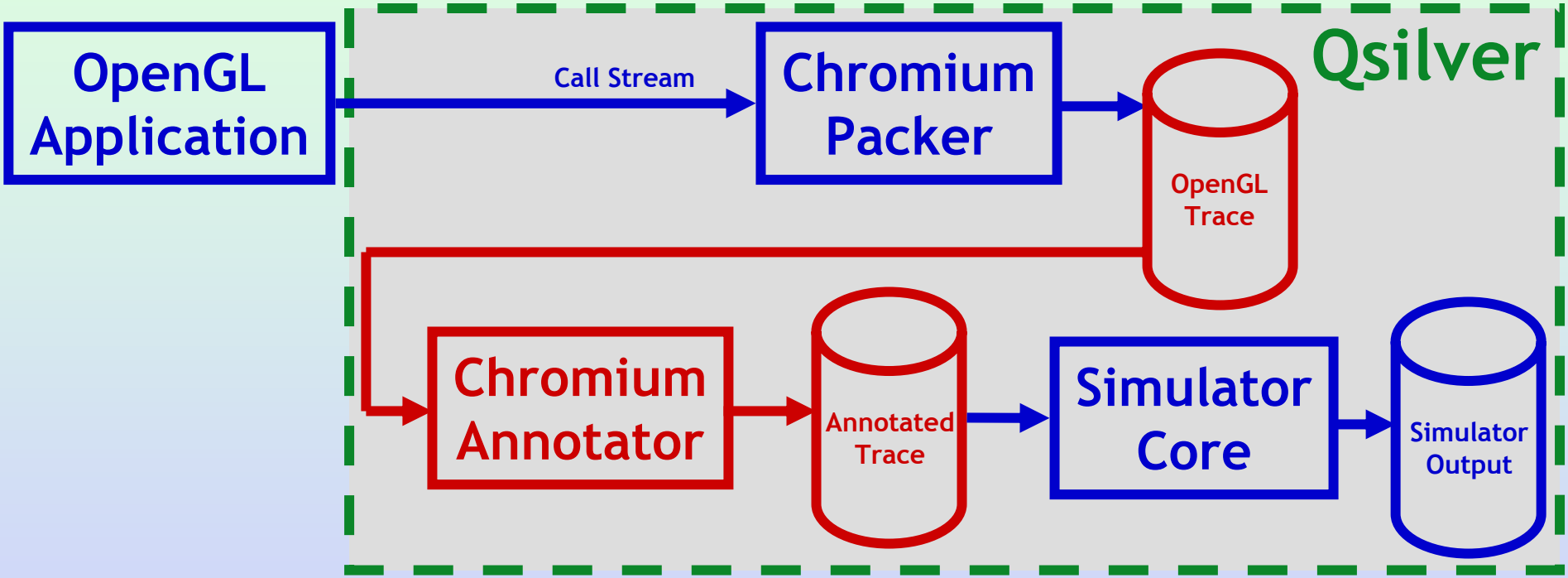
- An OpenGL application's call stream is intercepted by Chromium

Application to Simulation



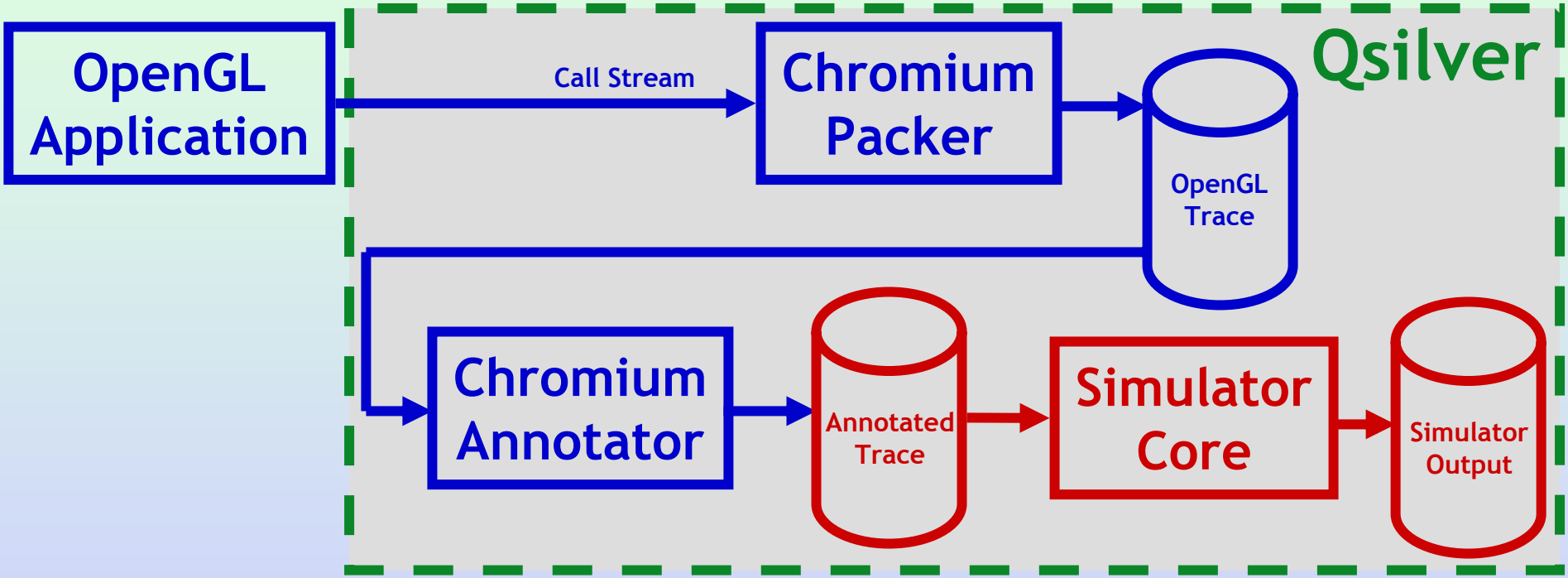
- The call stream is passed to the packer, which generates an OpenGL trace file

Application to Simulation



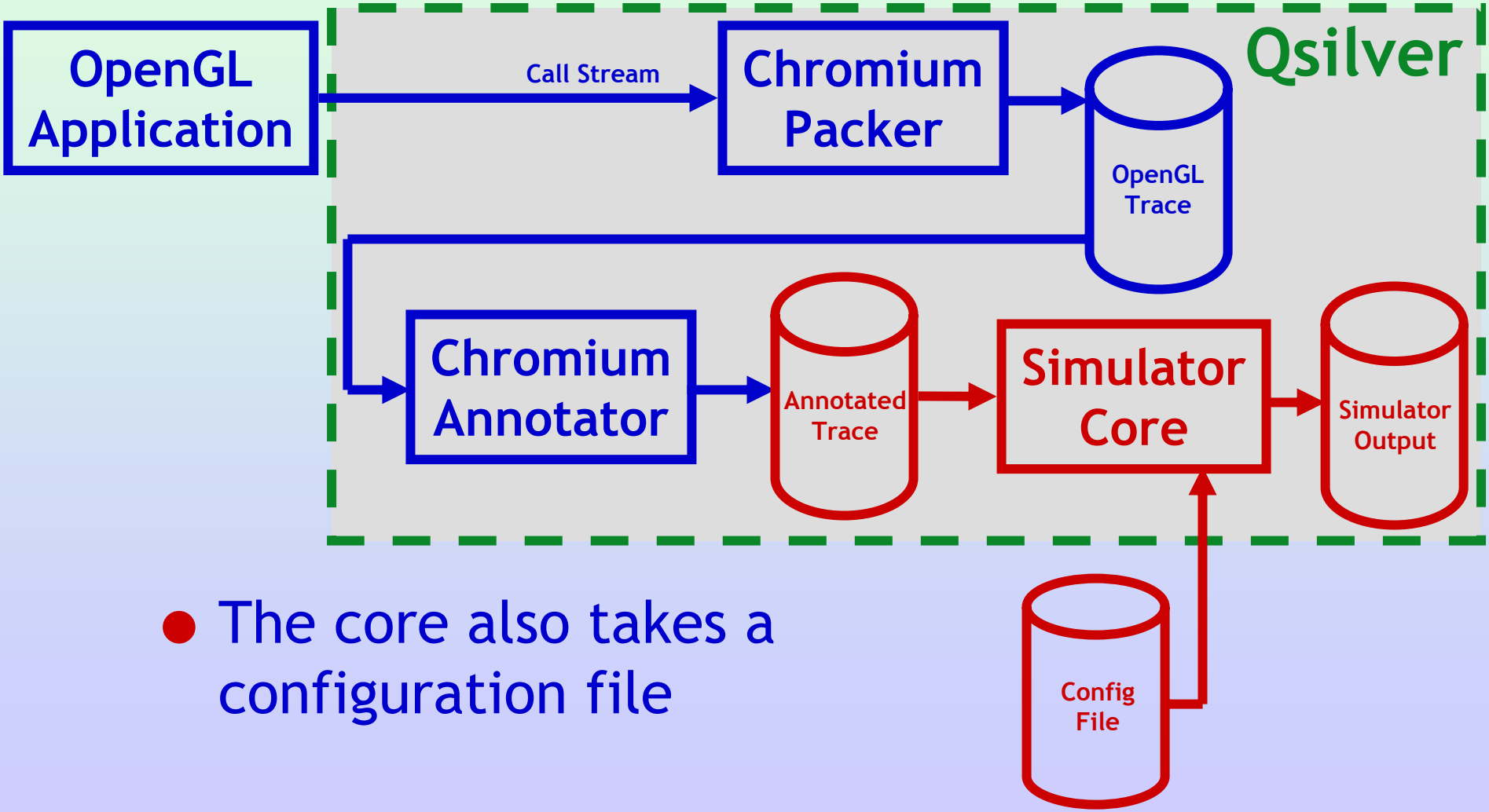
- The annotator reads the OpenGL trace and produces an annotated trace, which is the input to the simulator core

Application to Simulation



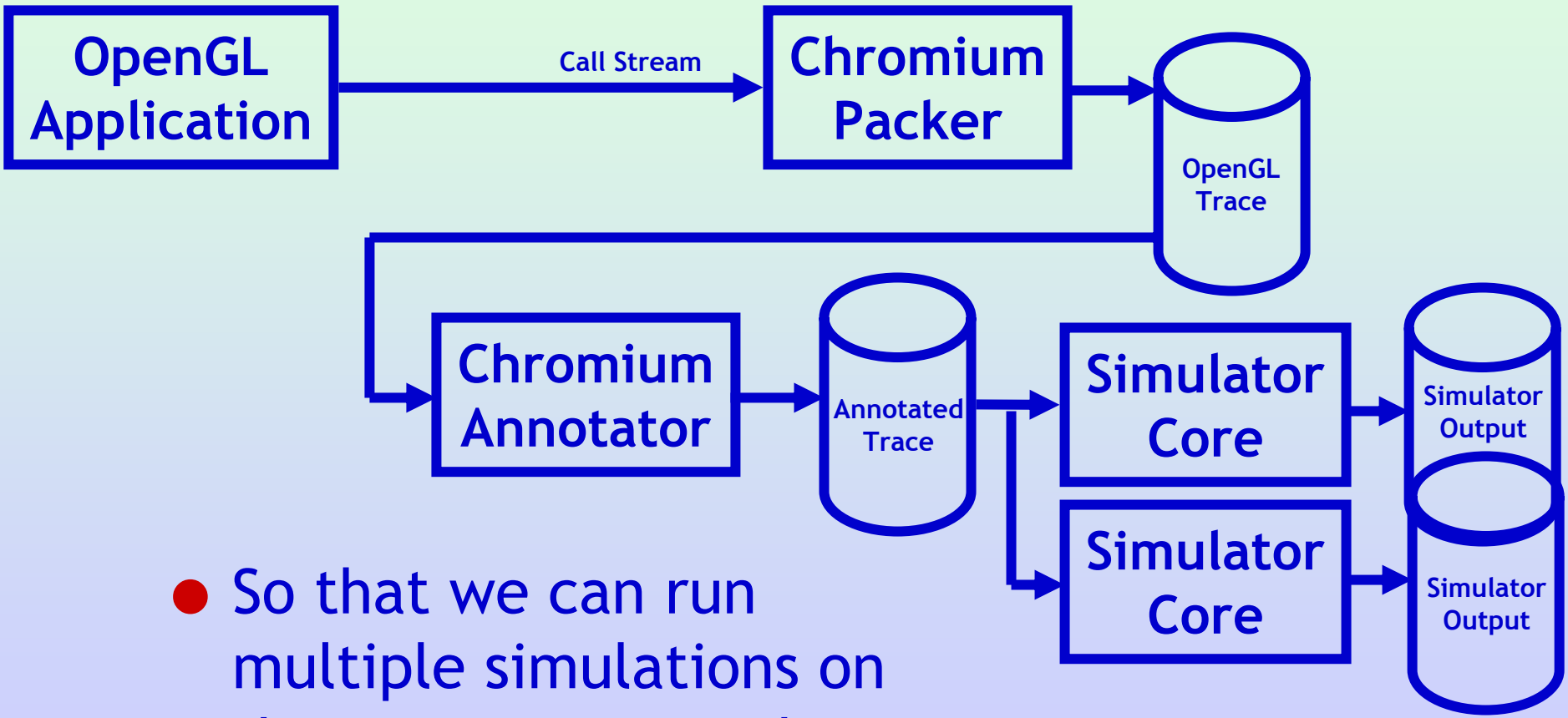
- The simulator core reads the annotated trace and produces the simulation results

Application to Simulation



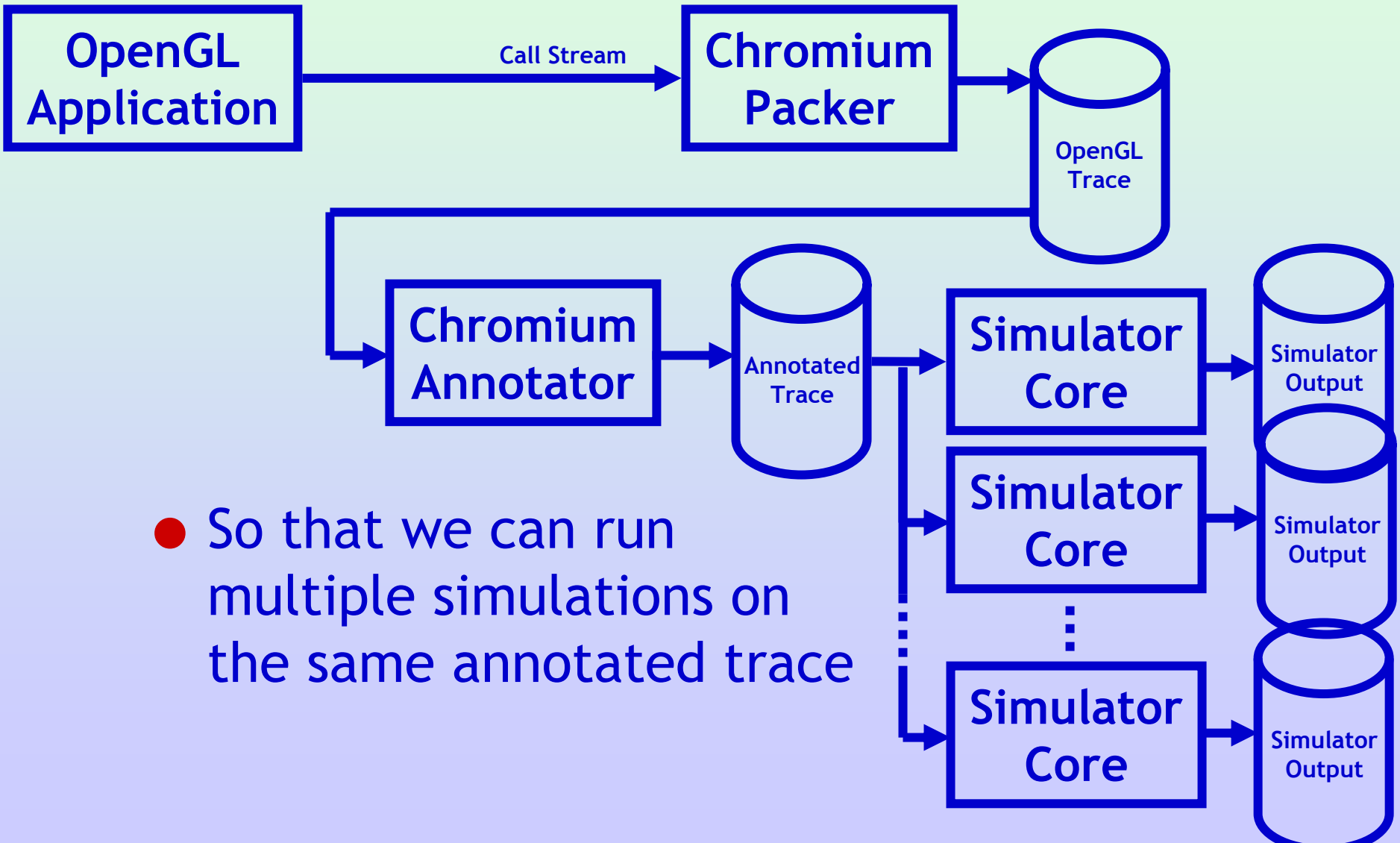
- The core also takes a configuration file

Application to Simulation



- So that we can run multiple simulations on the same annotated trace

Application to Simulation

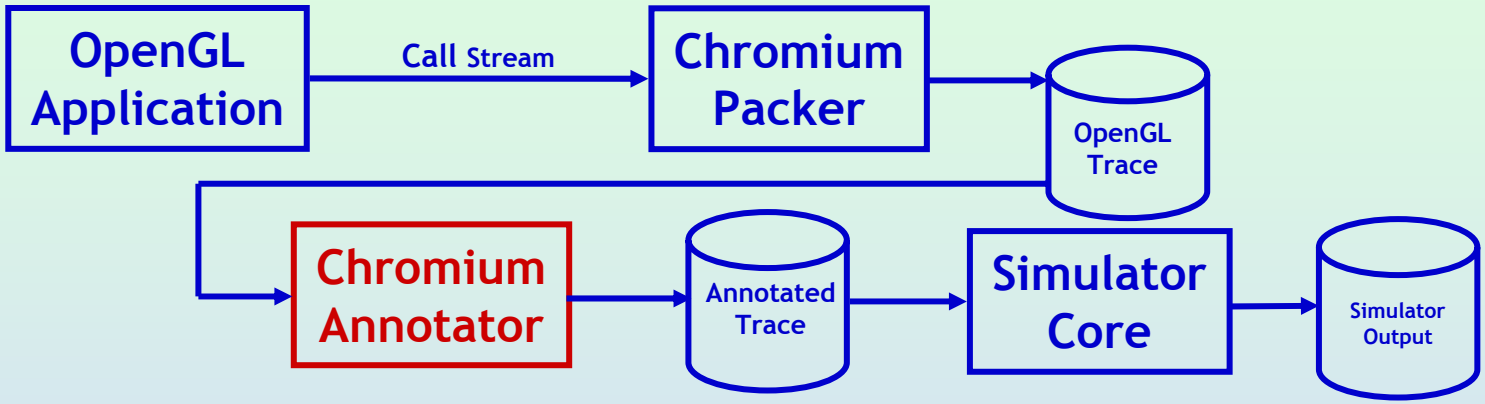


- So that we can run multiple simulations on the same annotated trace

Example: Counting Fragments

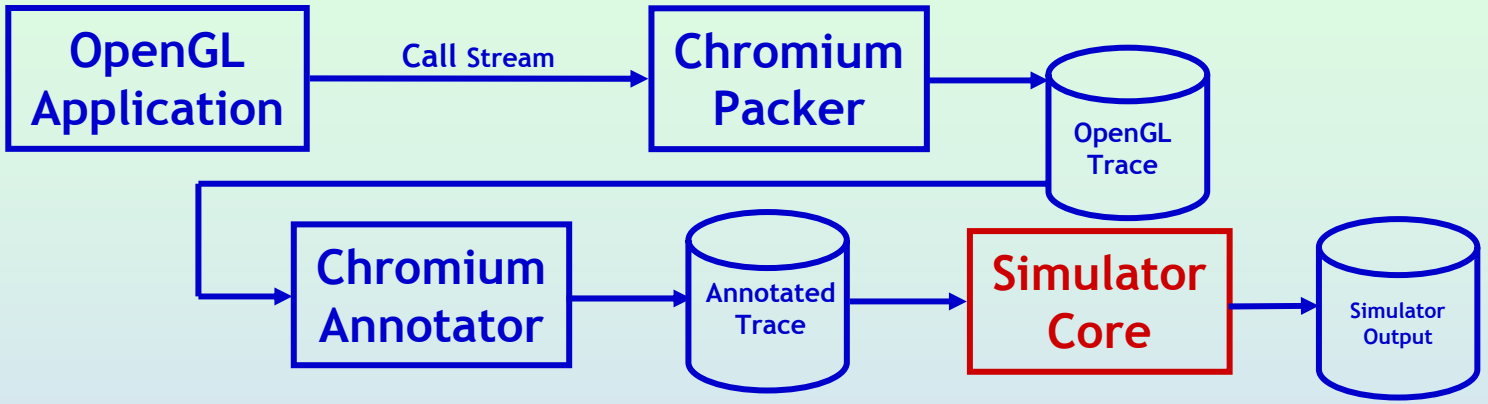
- OpenGL stream is transformed so that all geometry is rendered triangle by triangle
- Occlusion query wrapped around every triangle
- Two passes for every triangle
  - First: With depth buffer and depth test disabled
    - Counts all fragments generated
  - Second: With depth buffer and depth test enabled
    - Counts only fragments which pass depth test

Generating the Input Trace



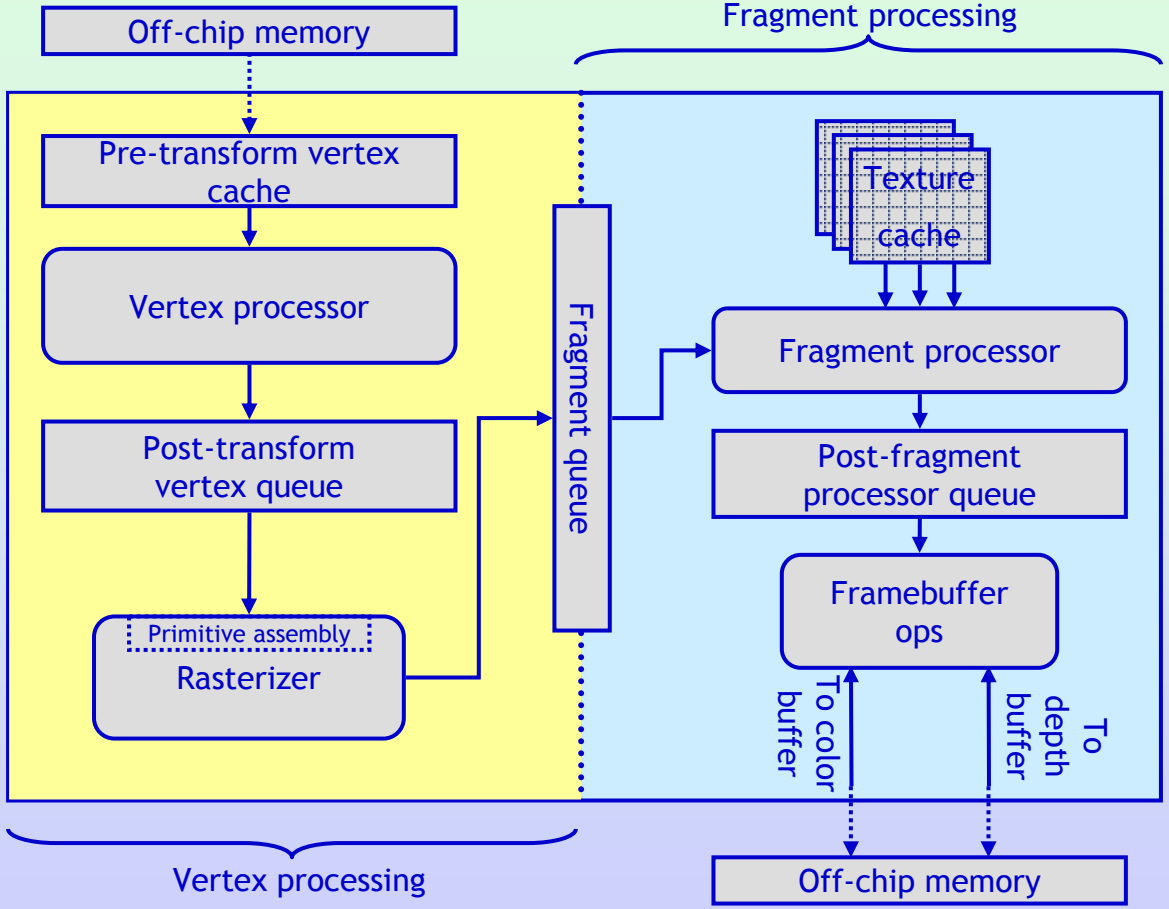
- Use similar Chromium transformations to gather for each triangle:
  - Number of fragments generated
  - Number of fragments Z-passed
  - Number of fragments on mipmap magnification filter
  - Number of texture accesses
  - Etc.

Cycle Timer Model



- Instrumented trace is the input to the simulator core
- Cycle timer is a *timing simulation*—no computation
  - Already know *what* events happen
  - Concern is to model *when* they happen

Architectural Model



- Our results are based on this hypothetical, fixed function pipeline
- Nothing precludes modeling more detail or adding programmability

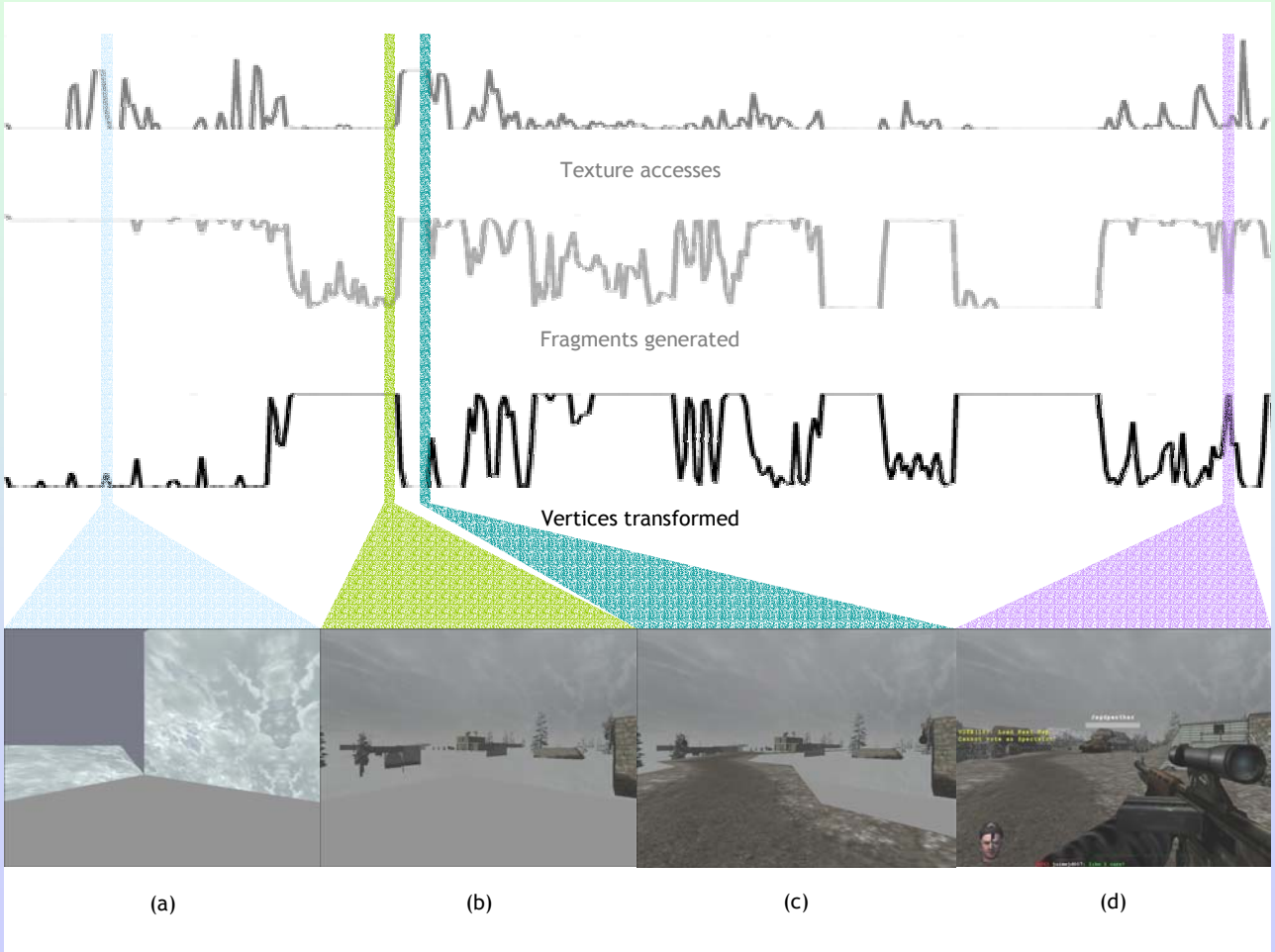
Modeling Power

- Qsilver power model based on an industry power model for a high performance CPU
  - Scale appropriately for voltage, frequency, semiconductor process, and bit width
  - Assume data-processing units are microcoded
    - Count events—vertices transformed, fragments created, etc.
    - Multiply by number of primitive operations per event (e.g., adds, multiplies, register/cache/FIFO reads…)
      - Estimates from NVIDIA fixed function pipeline code
    - Multiply by the power cost of a microcoded operation

Applications of Qsilver

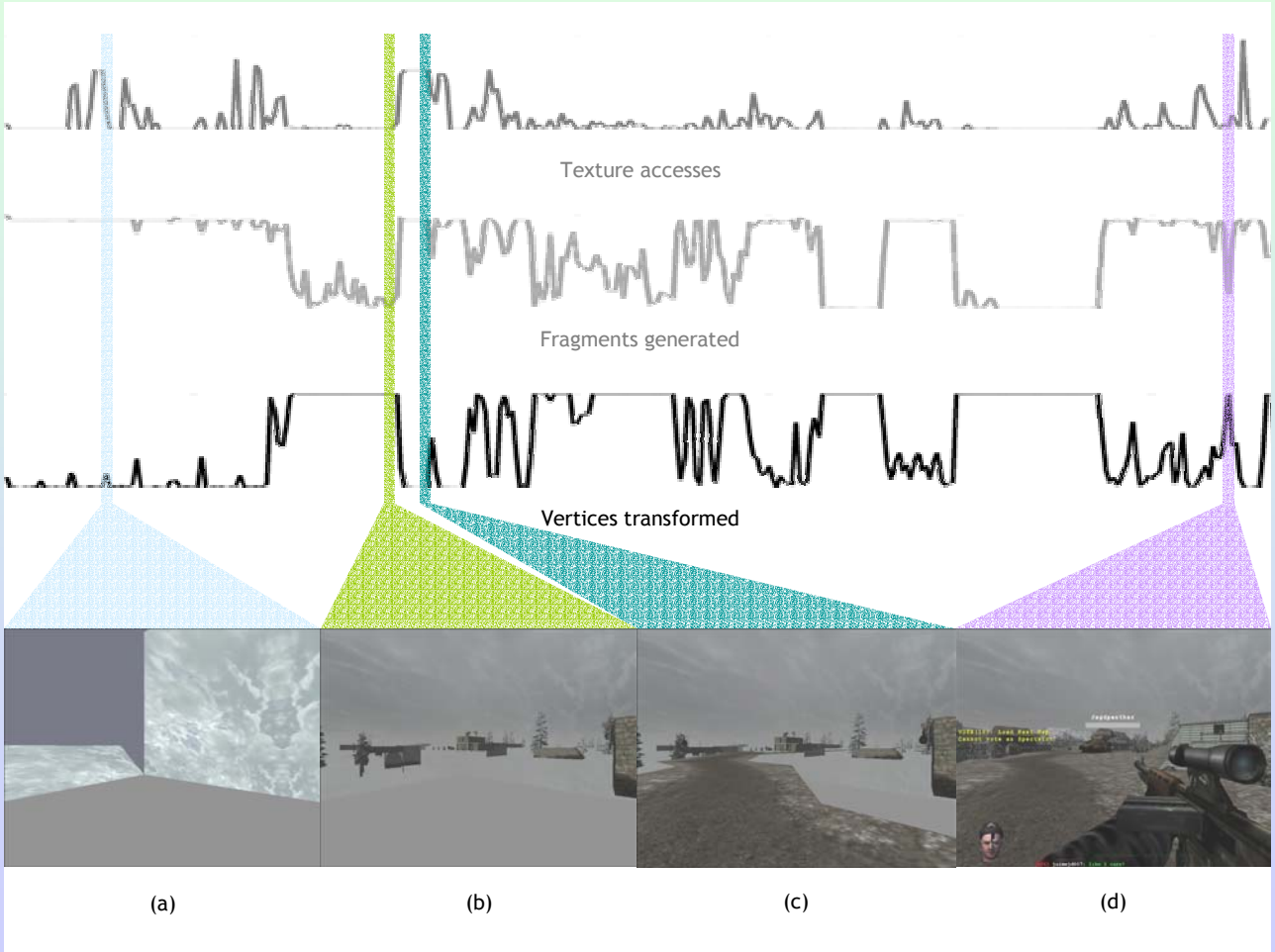
- We demonstrate Qsilver's applicability as a tool for
  - Performance analysis of OpenGL applications
  - Energy efficiency of graphics hardware
- We sketch how Qsilver can serve as a test platform for architectural features
  - For example, Z-min and Z-max culling

Performance Analysis



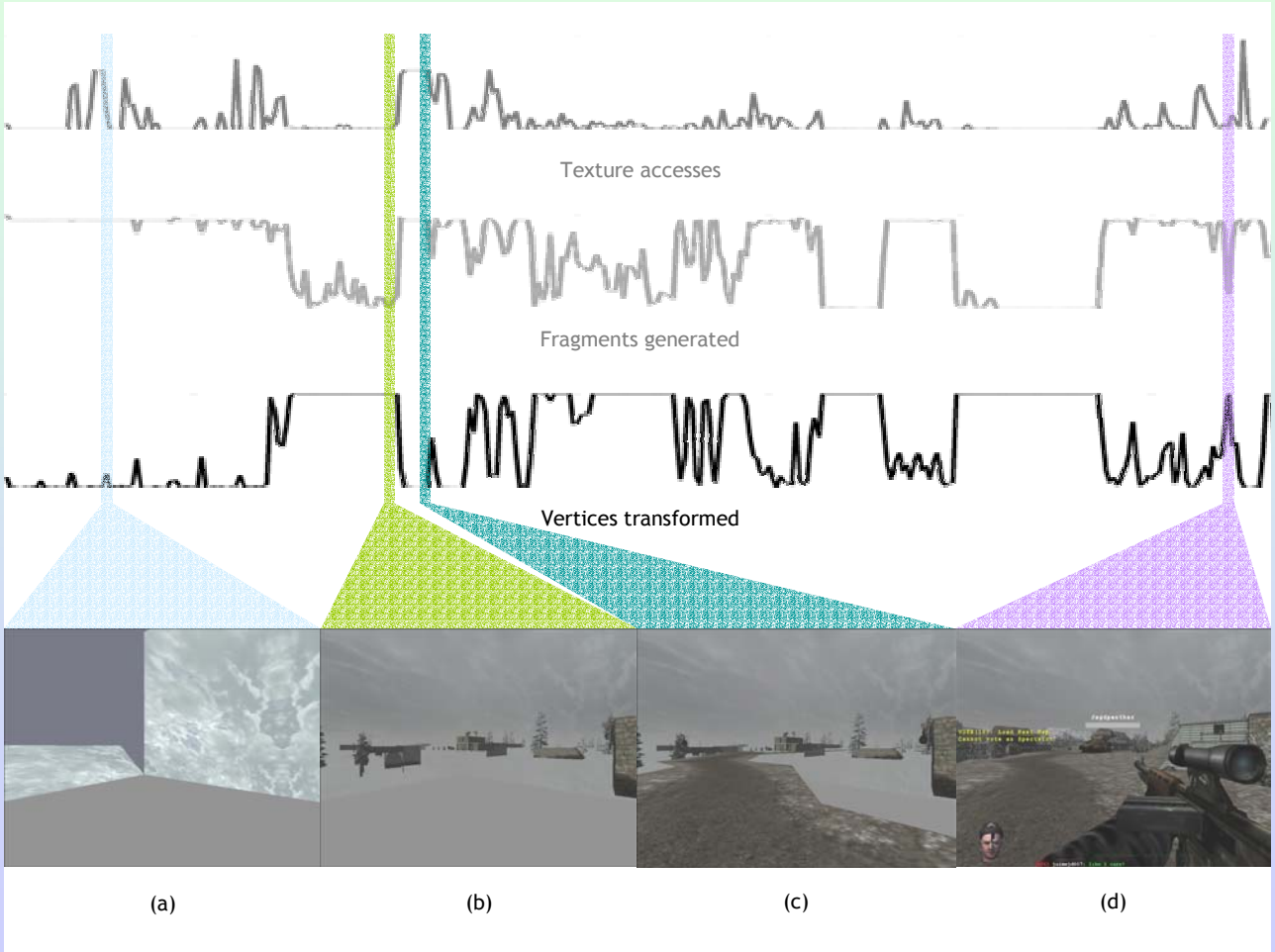
- In (a), the game fills in the textured sky-box

Performance Analysis



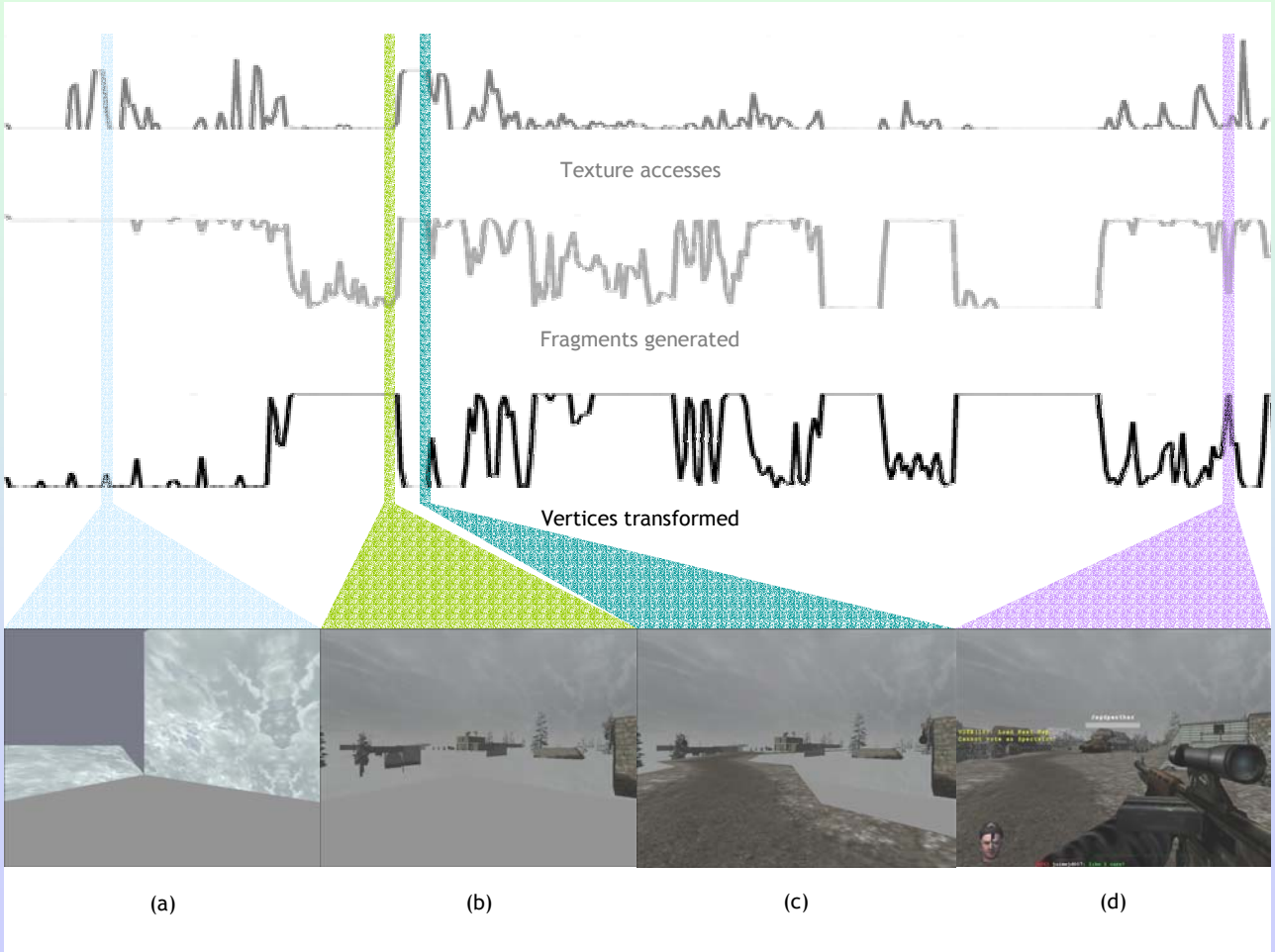
- In (b), the game moves on to details of trees and small buildings

Performance Analysis



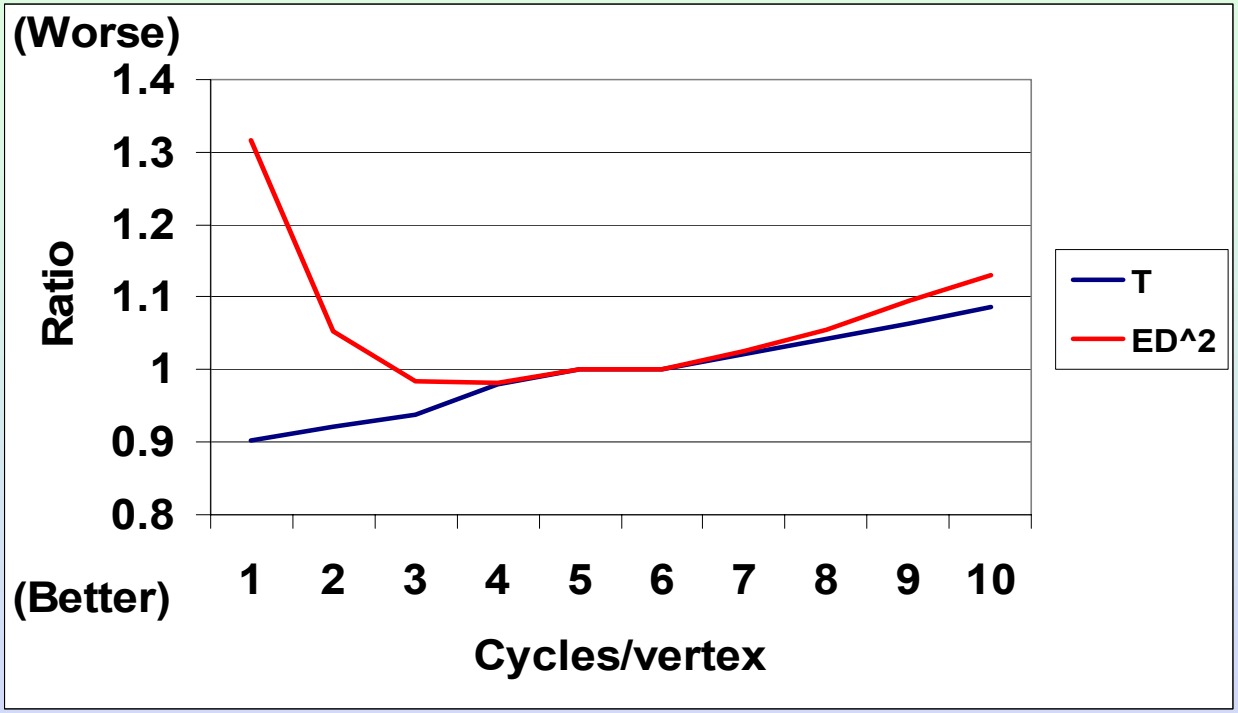
- (c) sees the placement of the road

Performance Analysis



- (d) adds the minutia of the face at lower left

Energy-efficiency Tradeoffs

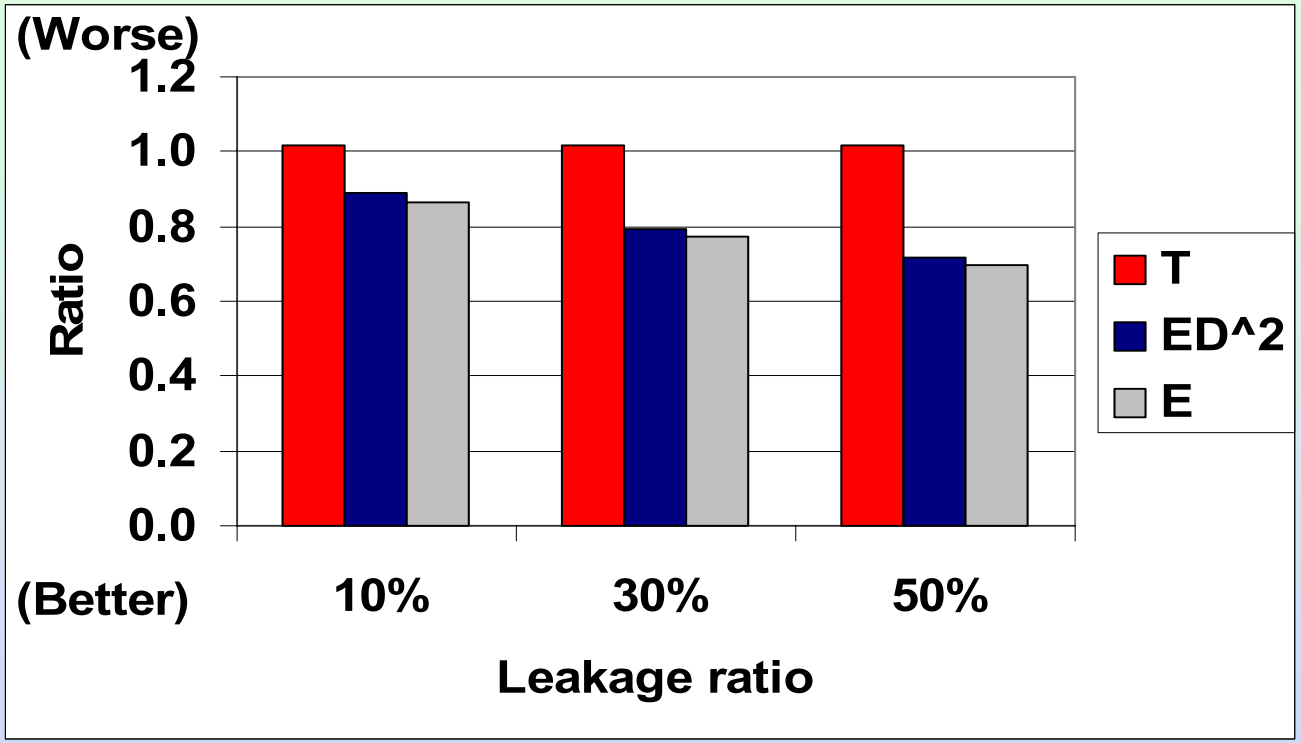


- Experiment: varying vertex throughput
  - T is performance
  - ED^2 is energy efficiency metric
  - All normalized to the unpipelined case
- ED^2 optimum is at 4 cycles/vertex

Multiple Clock Domains with DVS

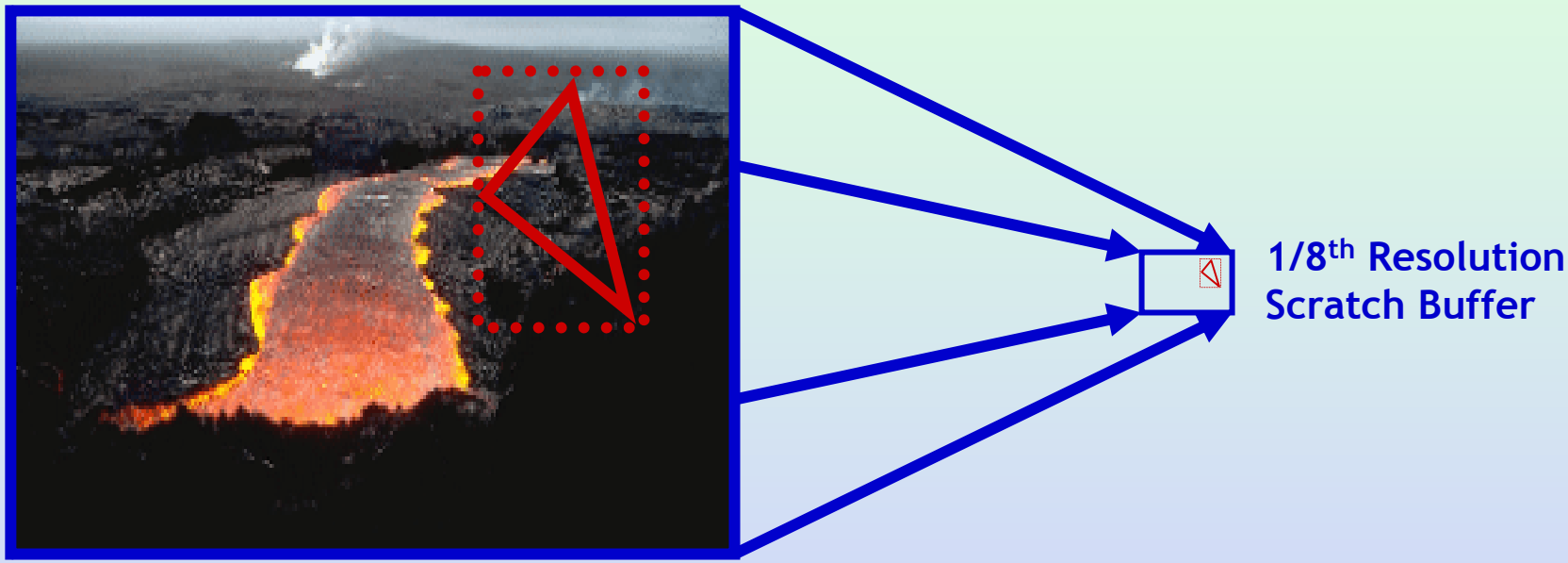
- Multiple independent clocks with dynamically scalable voltage
  - *Dynamic Voltage Scaling* (DVS) yields cubic reduction in power relative to performance loss ($P \propto V^2 f$)
- Takes advantage of the decoupling fragment queue
  - Pre- and post-fragment queue portions of the chip operate on independent clocks
- DVS setting is controlled by a simple state machine with hysteresis

Multiple Clock Domains with DVS



- Experiment: Multiple Clock Domains with DVS
  - T is performance
  - ED2 and E are energy efficiency metrics
  - All normalized to default case with no MCD
- The higher the leakage, the more DVS pays off

Z-Min Culling



- Render primitive in framebuffer and its filled bounding box in clear scratch buffer
- For each affected pixel in scratch buffer, find new min and max depth of corresponding block in framebuffer
- Re-render primitive in framebuffer with occlusion query and fragment program bound to count only fragments which pass Z-min test

Limitations

- Trace contains only aggregate information
  - No screen-space positions → hard to model:
    - Z-compression
    - Texture cache
    - Etc.
  - Chromium-based annotator makes non-aggregate data difficult to obtain
    - We plan to combine Chromium with Mesa to fill in the missing information

Conclusions

- Qsilver is a new framework for architectural simulation of GPUs
- Qsilver is flexible and highly configurable
- Demonstrated Qsilver's applicability as a tool for performance analysis and energy efficiency study
- Qsilver has the potential to stimulate graphics architecture research

Ongoing and Future Work

- Ongoing
  - Plug-in architecture with runtime pipeline configuration
  - Thermal simulations with *HotSpot*
    - Presented in poster at SIGGRAPH 2004
- Future
  - Prepare Qsilver for public release
  - Iterative refinement
    - Refine power model
    - Pipeline model
  - Collect more complete data in the input trace, including screen-space position

The End

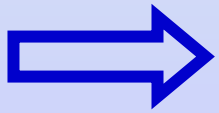
http://qsilver.cs.virginia.edu/

Vertex Arrays

- Store vertex arrays in memory
  - Never pass them to the renderer
- Replace accesses into a vertex array with immediate mode calls

```
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, 0, verts);

glBegin(GL_TRIANGLES);
glArrayElement(6);
glArrayElement(28);
glArrayElement(496);
glEnd();
```



```
glBegin(GL_TRIANGLES);
glVertex3fv(verts + 6 * 3);
glVertex3fv(verts + 28 * 3);
glVertex3fv(verts + 496 * 3);
glEnd();
```

Complex Geometries

- Potentially self occluding and not individual triangles
- Replace with equivalent set of triangles

```
glBegin(GL_TRIANGLE_STRIP);
glVertex3fv(verts);
glVertex3fv(verts + 1);
glVertex3fv(verts + 2);
glVertex3fv(verts + 3);
glEnd();
```

```
glBegin(GL_TRIANGLE);
glVertex3fv(verts);
glVertex3fv(verts + 1);
glVertex3fv(verts + 2);
glEnd();

glBegin(GL_TRIANGLE);
glVertex3fv(verts + 2);
glVertex3fv(verts + 1);
glVertex3fv(verts + 3);
glEnd();
```

Display Lists

- Potentially self occluding
- To handle:
  - Store GL trace in memory
  - Replay it when the list is called
    - Not baked in!
    - The renderer never sees the list as an object
- glCallList invokes the stored code

Counting Texture Accesses

- Check GL state for current texture mode for each triangle
  - Trivial multiplier for texture accesses per fragment
  - If any form of mipmapping is enabled
    - GL_MIN_FILTER and GL_MAG_FILTER require different number of texture lookups!
      - Bind fragment program to determine mipmap level
      - Render the triangle a third time with another occlusion query

Energy-efficiency Tradeoffs

- Highest performance and most energy-efficient design points typically not the same
- Use energy-delay-squared (ED^2) as energy-efficiency metric
  - Established metric in the low-power design community
    - Smaller $ED^2 \rightarrow$Better energy efficiency
  - Voltage independent

Performance Analysis

- We analyze a typical series of frames from Splash Damage's *Enemy Territory: Escape from Castle Wolfenstein*