Enabling Differentiable Rendering via AD of Parallel C++ STL Primitives in Clad

Midterm Evaluation

Student: Abdelrhman Elrawy

Mentors: Vassil Vassilev, Alexander Penev

Organization: HEP Software Foundation (CompRes/Clad)

Adding Concurrency and Inverse Rendering to Clad

Project Overview: To extend Clad's algorithmic differentiation capabilities to support multi-threading and path tracing applications.

Part 1: Concurrency (Forward-Mode AD)

- Propagating derivatives through reference wrappers.
- Adding differentiation support for std::thread.

Part 2: Inverse Rendering (Reverse-Mode AD)

- Resolving deep-cloning and loop-tracking compiler bugs.
- Building an initial soft-rasterization demo.
- Differentiating the traditional SmallPT path tracer.

Supporting Reference Wrappers (PR #1851)

Enabling std::ref and std::cref in Forward-Mode

Context

C++ threads copy arguments by default. Passing arguments by reference requires std::ref, but Clad needed to learn how to propagate derivatives through these wrapper objects.

Implementation

- Added custom pushforwards for std::ref and std::cref to generate derivative wrappers.
- Added support for .get() extraction to yield both primal and derivative values.

```
namespace std {
  // Instructs Clad on how to push derivatives through st
  template <class T>
  inline clad::ValueAndPushforward<::std::reference_wrappe
                                   ::std::reference_wrappe
  ref_pushforward(T& t, T& d_t) noexcept {
    // Returns a pair: primal reference and derivative re
    return {::std::ref(t), ::std::ref(d_t)};
  }
}
```

Adding Support for std::thread (PR #1891)

Forward-Mode AD for Concurrent Execution

Context

Using standard threading primitives previously caused Clad to generate invalid code or fail during the differentiation pass.

Implementation

- The AST visitor now recursively extracts the thread's callable payload and requests a derivative pushforward.
- Updates utils::isCopyable to detect deleted/private copy constructors, ensuring std::move semantics are used for non-copyable objects.

```
// 1. Primal C++ function dispatching work to a thread
double f_thread_ref(double x) {
  std::reference_wrapper<double> rx = std::ref(x);
  std::thread t(inc_ref, rx);
  t.join();
  return x;
}

// 2. Clad automatically generates the thread pushforward
int main() {
  auto d_thread = clad::differentiate(f_thread_ref, "x");

  // Evaluates both primal logic and derivative calculatio
  printf("thread: %.4f\n", d_thread.execute(2.0));
  return 0;
}
```

Compiler Fixes for Reverse-Mode AD (PR #1876)

Resolving AST Deep-Cloning and Reverse-Forward Pass Issues

Context

This PR addresses two reverse-mode AD stability issues in Clad's AST cloning and reverse-forward pass handling, adding regression tests to prevent crashes in SmallPT-like code.

Changes

- Deep-clone argument expressions for CXXConstructExpr and CXXTemporaryObjectExpr in StmtClone to avoid sharing/mis-owning subexpressions.
- Add a loop guard in reverse-mode call handling to disable restore_tracker-dependent reverse-forward behavior when no tracker is available.

```
// lib/Differentiator/ReverseModeVisitor.cpp
// Resolving tracker state bleed in reverse-mode loops

if (isInsideLoop && usingRestoreTracker && !m_RestoreTrack
  calleeFnForwPassFD = nullptr;
  usingRestoreTracker = false;
}
```

Initial Inverse Rendering Demo (PR #1789)

Differentiable Geometry via Soft Rasterization

Context

Traditional path tracing relies on hard intersections, which create discontinuities that can return zero or NaN gradients in AD tools.

Approach

Introduced a "soft rasterization" model as a first step. By replacing hard boundaries with a Gaussian falloff, the math became smooth enough for Clad to calculate gradients and optimize scene parameters (camera, light, and objects).

```
// Differentiable approximation: soft-weight Gaussian fal
// Allows gradients to flow smoothly by avoiding hard geo
double ray_sphere_proximity(...) {
  // ... distance math ...

  // Smooth version: use sd^2 regardless of sign for grad
  double sd = dist - r;
  return sd;
}

double soft_weight(double sd, double sharpness) {
  // Gaussian falloff acts as the differentiable workarou
  return exp(-sharpness * sd * sd);
}
```

Differentiating Traditional SmallPT (PR #1877)

Applying Native AD to the Original Path Tracer

Objective

To demonstrate that Clad can differentiate the traditional SmallPT path tracer natively, without relying on the soft-rasterization workarounds.

Implementation

- Modularized the original SmallPT codebase.
- Ran reverse-mode AD directly through hard intersections, recursive bounces, and Russian roulette path termination.
- Added custom reverse-mode pullbacks for vector math (SmallPTDiffCommon.h) to optimize compiler workload, and validated gradients using Finite Difference (FD).

```
// SmallPTDiffRadiance.h
// Native differentiation through traditional path tracing
inline Vec diff_radiance(Ray r, int depth, unsigned short
  // ...
  if (!diff_intersect(r.o.x, r.o.y, r.o.z, r.d.x, r.d.y, r
    return cl; // Hard miss

  // ...
  if (depth > 5 || !p) {
    // Clad successfully builds tapes through Russian rou
    if (sample_erand48(Xi) < p) {
      Vec scaled_f = f * (1.0 / p);
      f = scaled_f;
    } else {
      return cl;
    }
  }
}
```

Future Plan: 3DGS & Gather-Reduce

Transitioning from foundational capabilities to high-performance rendering optimization.

Phase 2: 3DGS Forward Redesign

Forking diff-gaussian-rasterization to strip out raw CUDA atomic loops.

- Implementing deterministic thrust::sort_by_key.
- Enforcing tile-based memory ownership via thrust::reduce_by_key to group pixel updates safely.

Phase 3: Clad Backward Synthesis

Triggering Clad's extended liveness analysis on the refactored renderer.

- Auto-generating the lock-free, zero-atomic backward pass from the Thrust abstractions.
- Validating gradients against finite differences and profiling memory throughput with Nsight Compute.