Project Title
Optimizing Mesh and Multires Sculpting Performance in Blender

Name
Namit Bhutani

Contact
Email: namit.bhut@gmail.com
Blender Chat: intellomaniac:blender.org
Blender Projects - https://projects.blender.org/Namit
GitHub: https://github.com/NamitBhutani
LinkedIn: https://www.linkedin.com/in/namitbhutani/

Synopsis

This project aims to improve the performance of Blender's sculpting system by implementing several targeted optimizations to the recently refactored sculpt brush system. The focus will be on reducing memory overhead, improving cache coherency, and optimizing the data processing pipeline for both regular mesh and multires sculpting workflows.

Benefits

The proposed optimizations will directly benefit artists by:

- Providing smoother, more responsive sculpting experiences, especially with high-resolution meshes.
- Reducing memory usage during sculpting sessions, enabling work on more complex models.
- Decreasing lag when working with multires sculpting at high subdivision levels.
- Potentially improving overall stability by optimizing memory management for undo operations.

Deliverables

1. Spatial Mesh Sorting Implementation

- I have already added a new operator for users to manually sort meshes.
- Implementation of modified IndexMask interface for BVH nodes.

- Need to do performance benchmarks testing improvements.

2. Local Data Copies Optimization

- Specialized handling for multires at high subdivision levels.
- Tuned chunk sizes for regular mesh operations.

3. Undo Data Format Improvements

- Implementation of compressed sparse arrays (will have to experiment with different approaches) for undo data.
- Non-blocking compression implementation (seems hard, but interesting to work on)

4. Multithreading Inside Nodes

- Enable multi-threading within nodes for expensive operations.
- I also want to work on a dynamic implementation which enables multi-threading within nodes when they’re above a certain vertex count.

5. Documentation

- Technical and User documentation on new features (mesh sorting operator).
- Performance profiling results.

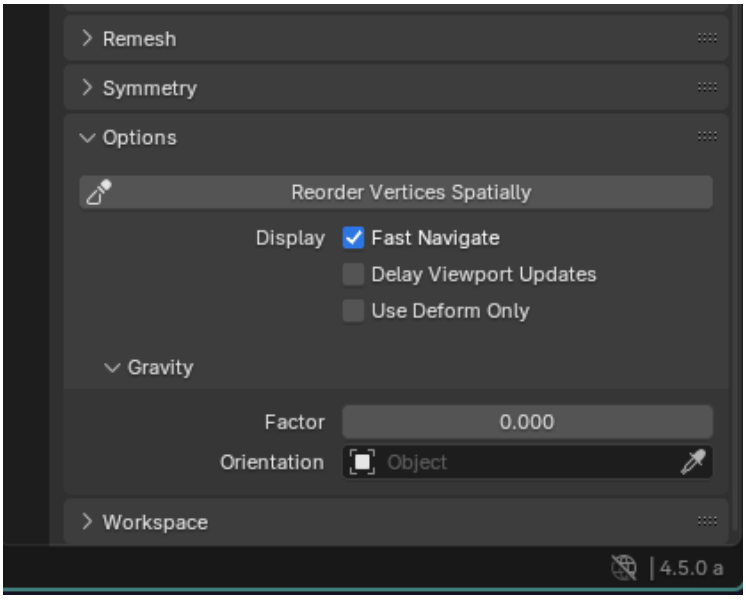
Project Details

Current Architecture and Bottlenecks

- **Data Organization:** Indirect access through index arrays causes additional memory lookups.
- **Undo System Overhead:** High memory usage for high-resolution sculpts.

Technical Implementation Details

1. Spatially Sort Meshes



- I have worked on a spatial sorting algorithm and in the pre period - I'll work with maintainers to finalise it.
- I have implemented the Mesh Reorder operator - I will work on finalising the flow of triggering the operator (before/after entering sculpt mode).
- Modify the BVH node interface to provide IndexMask instead of index spans.

2. Local Data Copies Optimization

- We can utilize the inherent contiguous memory of high subdivision multires faces to bypass useless array creation.
- Implement benchmark-based chunk size tuning for different multires subdivision levels.

3. Multithreading Inside Nodes

- Implement dynamic multithreading within leaf nodes based on number of nodes and size of nodes.
- I'll also keep in mind to implement robust task isolation to prevent conflicts while multi-threading.

4. Undo Data Format

- I will experiment with different ways to implement compressed sparse arrays for storing undo data such as COO, CSR, CSV formats.

- Then I will try to create a non-blocking compression system. (Benchmark several approaches like Double Buffered, Tiered Compression or Background Worker Thread)

Project Schedule

Community Bonding Period (May 8 – June 1)

- **Pre-Period Progress:**
    - Present initial spatial mesh sorting code.
- **Weeks 1-2:**
    - Work with maintainers and developers to review the current work.
    - Finalize a concrete implementation plan for spatial mesh sorting.
- **Weeks 3-4:**
    - Finalize the spatial mesh sorting plan based on feedback.
    - Begin full implementation of the spatial mesh sorting feature.

Coding Period (June 2 – August 25)

Phase 1: Spatial Mesh Sorting Implementation

- **Weeks 1-2 (June 2 – June 15):**
    - Complete the full implementation of spatial mesh sorting.
    - Undergo code review, testing, and documentation.

Phase 2: Local Data Copies Optimization for Multires Meshes

- **Weeks 3-4 (June 16 – June 29):**
    - Begin work on optimizing local data copies for multires meshes.

    - Implement initial changes and conduct preliminary tests using different chunk sizes.

- **Weeks 5-6 (June 30 – July 13):**

    - Continue optimizing, get the code reviewed, and refine documentation.
    - Finalize the Local Data Copies Optimization feature and add necessary documentation.

Phase 3: Multithreading and Undo Compression Experimentation

- **Weeks 1-2 (July 14 – July 27):**

    - Start integrating multithreading within nodes to optimize expensive operations.
    - Simultaneously, begin experimenting with undo data format compression.
    - Conduct initial performance benchmarking and code review.

- **Weeks 3-4 (July 28 – August 10):**

    - Refine the multithreading integration based on benchmark and feedback and add documentation.
    - Finalize the experimentation on undo compression, if time permits during GSoC, otherwise, document the findings and plan to continue post-GSoC.

- **Week 5 (August 11 – August 17):**

    - Perform bug fixing, comprehensive performance benchmarking, and integrate any remaining sub features.

- **Week 6 (August 18 – August 25):**

    - This will mainly be the buffer week, to cover up on any lost time and work.

Final Week (August 25 – September 1)

- Finalize all code contributions.
- Create a demonstration showcasing the performance improvements achieved.
- Prepare the final submission.

Bio

I am Namit Bhutani, a 21-year-old Computer Science undergraduate from BITS Pilani, Hyderabad, India, originally from Delhi. My journey with Blender began at the age of 14 with Blender 2.8 when I used it to model foliage assets for my 3D world in Unreal Engine 4. Over time, I revisited Blender with a programmatic approach—most recently exploring the capabilities of geometry nodes, which I find both innovative and super fun.

Parallel to my work with Blender, I developed a passion for programming, starting with web development before venturing into lower-level graphics programming. A few months ago, I began diving into Blender's source code. After setting up the development environment, I contributed by addressing issues in the community's Issues tab, eventually leading me to this GSoC project.

With guidance from Hans Goudey, I have been working on the Spatial Sorting operator, which has deepened my interest in contributing to Blender.

In addition to my Blender work, I am involved in a research project focused on simulating the interaction of polarized light with surfaces, which is still in the early stages. I have also built a basic OpenGL software rasterizer to further hone my understanding of graphics, and I challenged myself by creating a procedural landscape generator using OpenGL, compute shaders, and the marching cubes algorithm.

Through Google Summer of Code, I aim to make meaningful contributions to Blender, optimize its performance to benefit artists worldwide, and establish connections in the open-source community.