# Bullet & Houdini 13

Cameron White
cwhite@sidefx.com

SIDE EFFECTS SOFTWARE

# Overview

- New object representation (RBD Packed Object)
  - Improved performance and reduced memory usage
  - Can handle large numbers of objects very efficiently
- Constraint Networks
  - Procedurally create constraints of any type

# Packed Primitives - Overview

- New primitive type in Houdini 13
- Has a transform and a reference to geometry somewhere else (e.g. on disk or in a SOP network)
- The packed geometry is not necessarily in memory, so cannot be directly edited – must unpack it first (Unpack SOP)
- Convenient for RBD – can represent a set of primitives with a single primitive, and can perform efficient instancing
- Several types of packed primitives

# Packed Geometry Primitive

- Packs away the input geometry (Pack SOP)
- Any copies of the packed primitive share the packed geometry
- Ideal for instancing
- Copy SOP can also pack the geometry before copying
- Can change the viewport LOD – e.g. display only points or the bounding box

SIDE EFFECTS SOFTWARE

## Packed Fragment Primitive

- Similar to the Packed Geometry primitive, but references a **subset** of the primitives in the geometry
- Ideal for fracturing – for each fractured piece, we can have a single packed primitive that references the polygons of that piece
- Use the Pack SOP or the Assemble SOP

SIDE EFFECTS SOFTWARE

## RBD Packed Object - Overview

- Import geometry containing packed primitives from a SOP network
- Each packed primitive corresponds to a single object in the simulation
- The Bullet Solver uses the packed geometry when creating the object's collision shape
- Point attributes are created to store the velocity, mass, etc. for each object
- Less performance overhead and memory usage versus having a separate DOP object for each RBD Object.

# Fractured Object Shelf Tool

- Uses the Assemble SOP to create a packed fragment for each unique value of the 'name' primitive attribute
- If there isn't already a 'name' attribute, creates it based on connectivity
- Example: fracturing into ~1000 pieces
  - RBD Packed Object (H13): 262ms per frame, max memory usage of 735MB
  - RBD Fractured Object (H12.5): 713ms per frame, max memory usage of 1.5GB

## Point Object Shelf Tool

- Creates a Packed Geometry primitive for the object, and copies it onto each point of some source geometry
- Only one copy of the object's geometry is made!
- The Bullet Solver can detect when multiple Packed Geometry primitives reference the same geometry, and can make several optimizations to improve performance in such cases
- Example: pile of ~2000 boxes
  - 215ms per frame with the RBD Packed Object, memory usage: 350MB
  - 502ms per frame with the RBD Point Object, memory usage: 1.1GB

# Performance of Larger Simulations

- The RBD Packed Object represents large numbers of objects in a much more efficient manner than the RBD Fractured Object DOP or the RBD Point Object DOP
- Example: dropping ~70,000 objects for 10 frames:
  - RBD Packed Object (H13): 23.7s, memory used: 1.1GB
  - RBD Point Object (H12.5): 6m 52s, memory used: 8.9GB

# Using a SOP Solver

- To modify attributes of the objects, run a SOP Solver on the RBD Packed Object's geometry.
- For example, modify the 'active' point attribute to switch an object from active to static.
- Can add or remove packed primitives to create or destroy objects

- Each piece of an RBD Packed Object has a point with attributes such as 'P', 'v', 'mass', 'orient', etc. - this is very similar to a particle system
- In Houdini 13, particles are in DOPs, so we can use POP nodes to manipulate objects in a Bullet simulation!
- Bullet Solver recognizes point attributes such as 'force', 'torque', 'targetv', and 'airresist', and can use them to apply forces to each object
- Example: POP Curve Force

# DOP Import SOP

- New modes of operation to support packed primitives and RBD Packed Objects
  - “Fetch Unpacked Geometry from DOP Network” will unpack any packed primitives in the geometry
  - “Fetch Packed Geometry from DOP Network”
    - Creates a packed primitive for each object being fetched from the DOP Network
    - Directly returns the packed primitives of any RBD Packed Objects
    - Packs the geometry of any regular RBD Objects

# Limitations

- Currently cannot interact with other solvers (fluids, cloth, etc.)
- Not compatible with some RBD constraint DOPs (such as the RBD Spring Constraint DOP)
  - These constraints work based on DOP objects, and cannot reference an object that is part of a packed object
  - Use constraint networks instead (see next slides)

# Constraint Networks

- Generalization of the Glue Network Constraint from Houdini 12.5
- Uses geometry as a template to construct constraints
- Much simpler and faster for procedurally creating constraints
  - For example, creating springs between adjacent bricks in a wall
- Can construct any type of constraint that the Bullet Solver recognizes (glue, spring, pin, slider, or cone twist)
- Works with both RBD Packed Objects and regular RBD Objects

# Constraint Network Geometry

- Each point represents an anchor of a constraint
  - The 'name' attribute identifies which object to attach the anchor to
    - For RBD Packed Objects, the 'name' point attribute is used
    - For regular RBD Objects, the DOP object name is used
    - If the name is invalid, the constraint will be attached to a world space position
  - 'P' attribute specifies the initial world space position of the anchor
  - For rotational anchors, 'r' specifies the initial orientation
  - Several other attributes can be used – see the Constraint Network DOP's help card

# Constraint Network Geometry

- Each primitive represents a single constraint, and is a line that connects two points (the anchors)
  - The 'constraint_name' attribute specifies the Data Name of a constraint that was attached to the Constraint Network DOP
  - The 'constraint_type' attribute specifies whether the constraint affects 'position', 'rotation', or 'all' degrees of freedom
- The subdata referenced by 'constraint_name' provides the **default values** for any attributes of the constraint (such as strength or damping)
- If there is a primitive attribute with the same name (e.g. 'strength'), the default value is **multiplied** by the value of that attribute

# Constraint Network DOP

- Attach the RBD constraints that can be created (e.g. a Spring Constraint Relationship DOP)
- Specify SOP geometry that defines which of those constraints to create between each pair of objects.
- Wire in SOP Solvers to modify the constraint network on each frame
- Example: Glue Adjacent shelf tool

# Connect Adjacent Pieces SOP

- New SOP for creating constraint network geometry to be used with the Constraint Network DOP
- Uses new VEX geometry creation functions ('addpoint', 'addprim', and 'addvertex')
- Two modes of operation:
  - Connect each point to the closest point from another piece
    - Use the “Max Connections” parameter to allow multiple connections per point
  - Create connections between the centroids of nearby pieces
    - Use the “Offset From Centroid” parameter to move the point towards the surface
- Performance tip: begin with small values for Search Radius and Points per Area, then increase as needed

# Modifying A Constraint Network

- Use the “Overwrite with SOP” parameter to re-import the constraint network geometry on specific frames
  - Allows for completely animated constraint behaviour, such as deleting constraints on specific frames
  - Example: Controlled breaking of glue bonds
- Or, use a SOP Solver to modify the constraint network on each frame
  - Useful for modifying the constraint network based on events in the simulation

# Using a SOP Solver

- Wire SOP Solver(s) into the last input of the Constraint Network DOP
- Deleting a primitive will delete the corresponding constraint from the simulation
- The 'broken' primitive group contains any constraints that were broken by the Bullet Solver itself (currently, this only happens for glue bonds)
  - This can be used to trigger events (for example, emitting debris) when a glue bond is broken
  - The solver will ignore any primitives in the 'broken' group on future frames

# Using a SOP Solver

- Several primitive attributes are updated by the solver to contain the current state of the constraint
  - The 'force' attribute specifies the force that was applied to satisfy the constraint
  - The 'distance' attribute specifies the distance between the two anchors
  - For glue bonds, the 'impact' attribute stores the accumulated impulse
- Example: Break springs that have stretched too far

# Using a SOP Solver

- The 'constraint_name' and 'constraint_type' primitive attributes can be modified during the simulation to change the type of a constraint
- Example: Change spring constraints into glue bonds

SIDE EFFECTS SOFTWARE