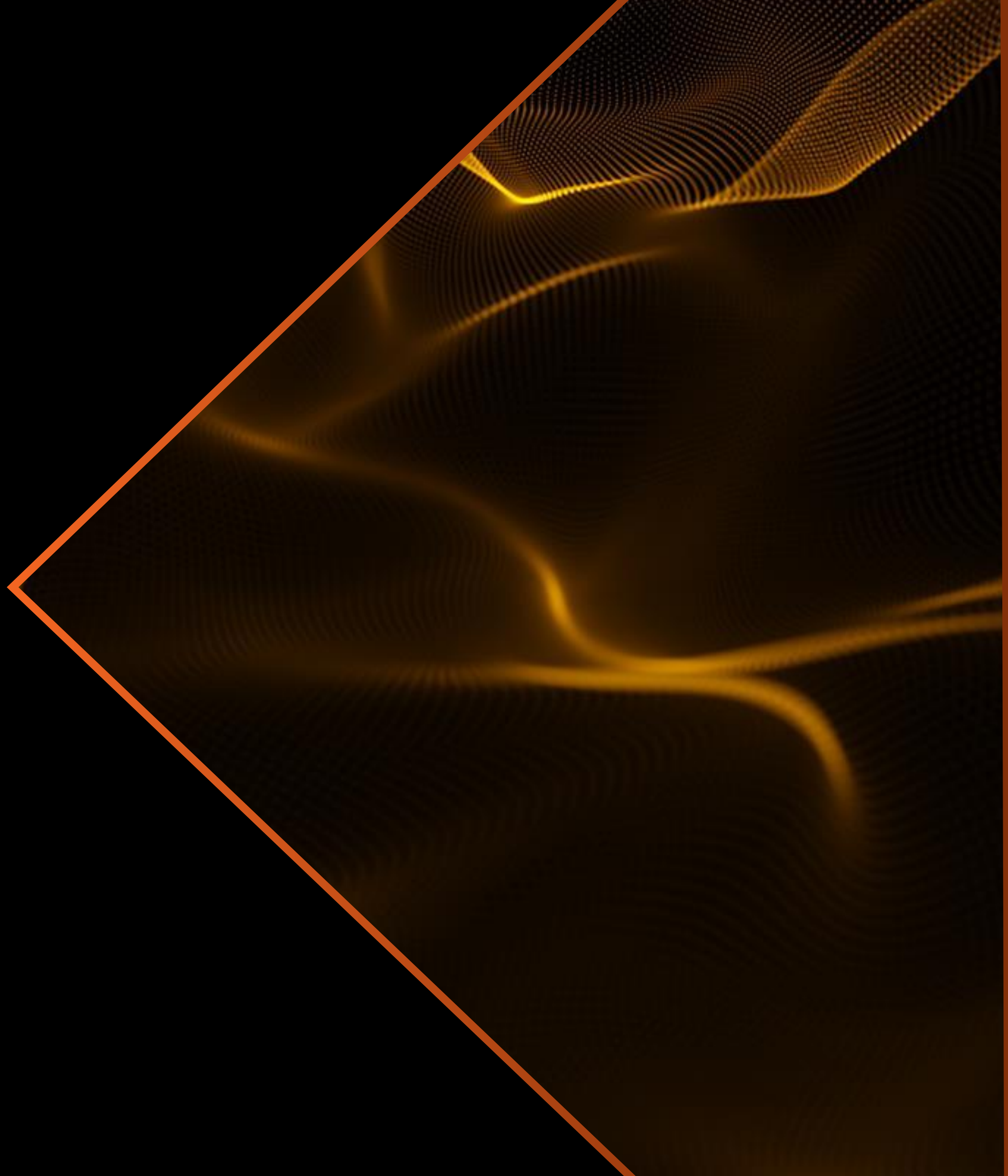
Developing an Unreal Plugin for FEM Physics

Eric Larsen

AMD Immersive Technologies Team

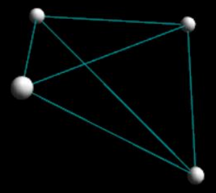


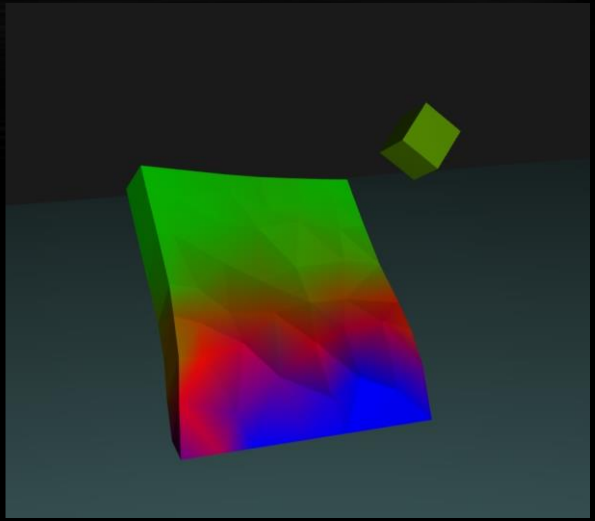
Talk Objectives

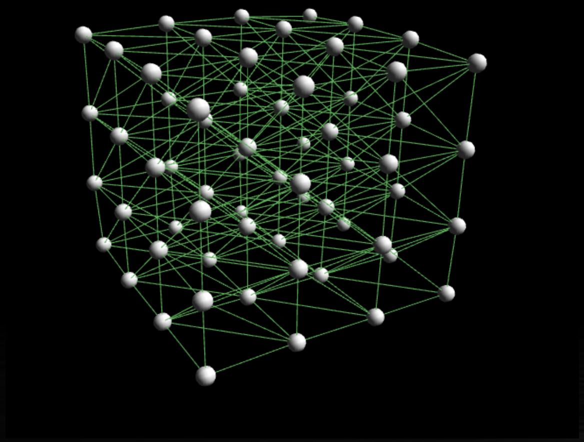
- Introduce the FEM R&D we've done
- Show you how we leveraged UE4's extensibility by creating a custom plugin
  - Custom rendering of FEM objects
  - Custom uassets and scene editing
- Show some demos!

FEM Fundamentals

- Physics method for deformable materials
- Models a solid as a mesh of elements (tetrahedra)
- Each element has material parameters
  - Stiffness
  - Volume preservation
  - Stress limit before permanent (plastic) deformation
  - Stress limit before fracture

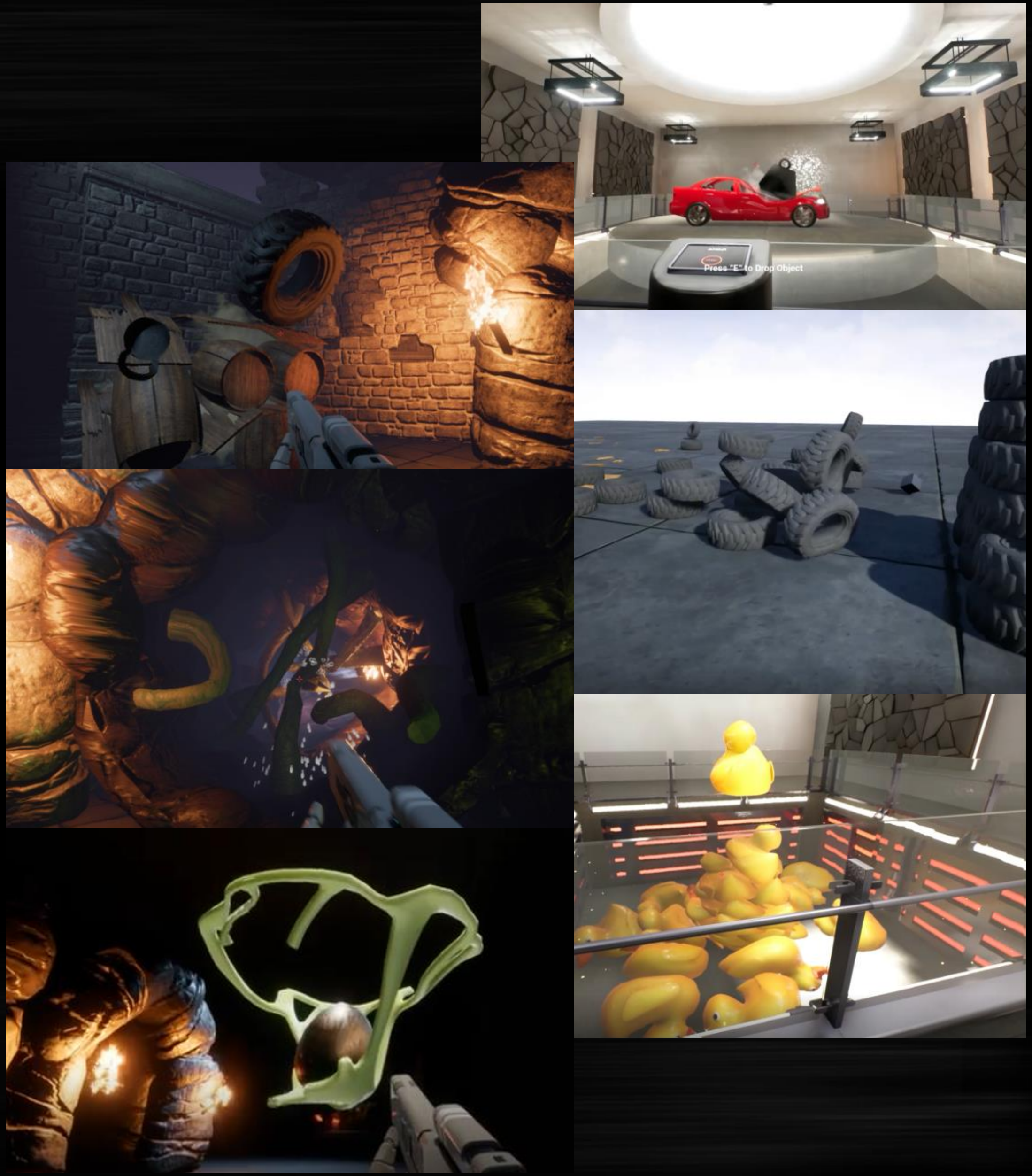




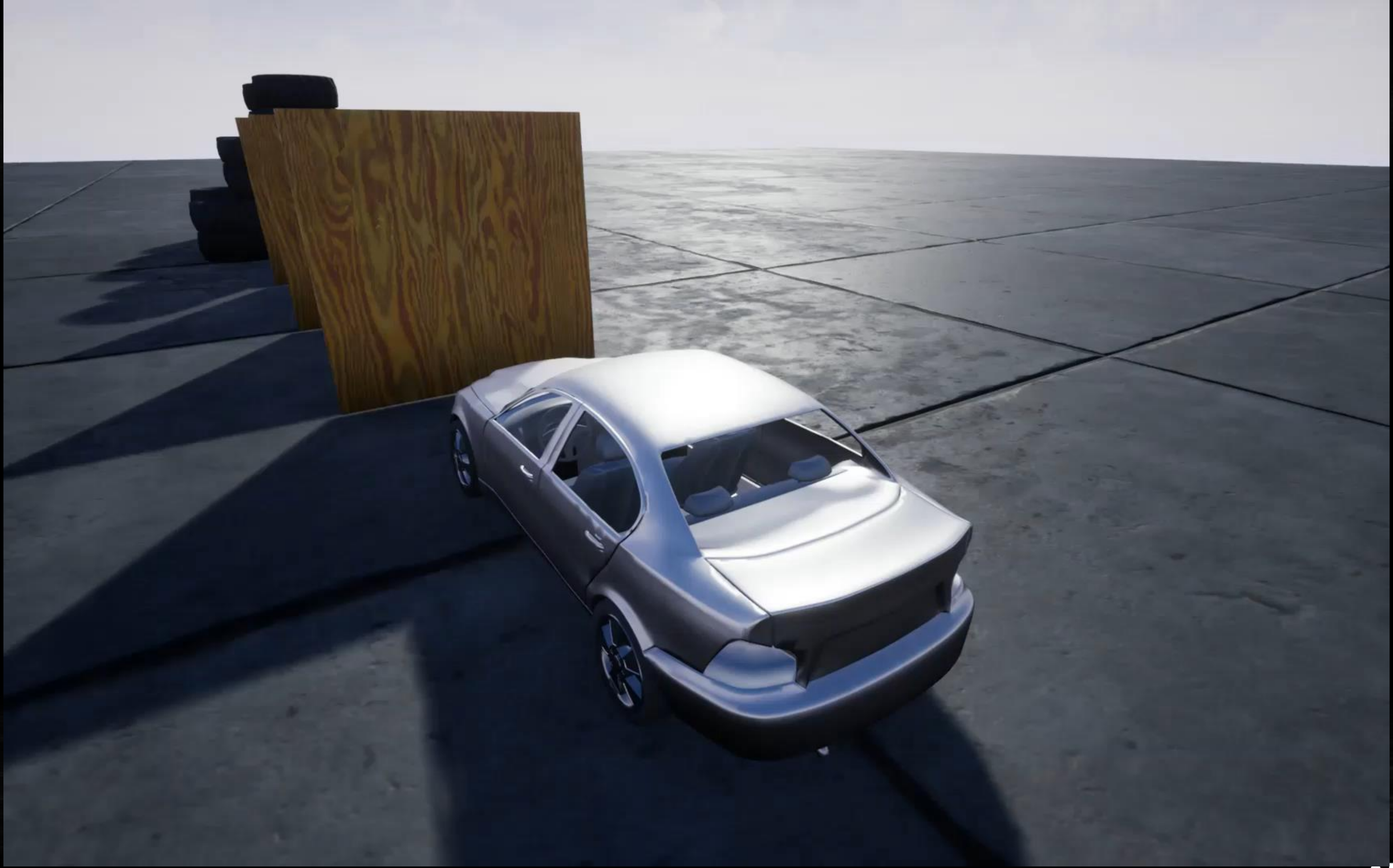


Applications

- A different kind of physics for different effects
- Deformable materials
  - Bending metal
  - Wood that flexes and breaks
  - Deforming, bouncing tires
  - Melting objects
  - “Alien” materials
- Fidelity
  - How objects flex, oscillate
  - Realistic material interactions
  - How things absorb and release energy, snap
  - Less brittle look
- New types of interaction
- Other examples/inspiration:
  - DMM middleware, MPC’s Kali



Press "E" to Drop Object



FEMFX Library

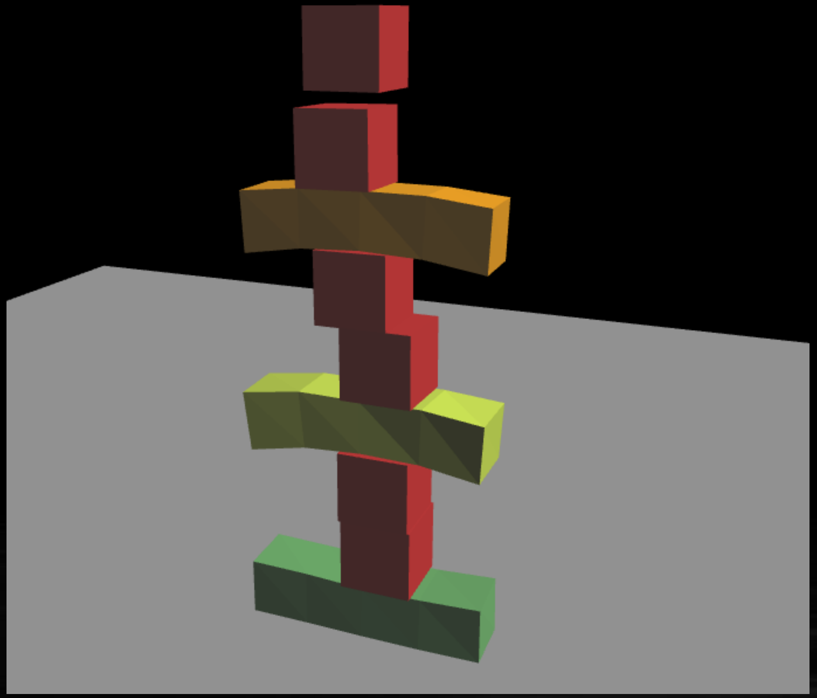
- Mix of published methods, shortcuts and custom solutions
- Multithreaded CPU implementation
    - For interaction with gameplay and other systems
    - Following trend of increasing CPU cores
    - Many SIMD optimizations

Method Highlights

- Implicit integration for stiff materials
  - Stability
  - Solves object vertices together
- Constraint-based contact between objects
  - Stacking
  - Compatibility with rigid-body solvers
- Continuous collision detection
  - Reduced pass-through
  - Iterative method
- Multiplicative Plasticity
  - Found to behave better than additive
  - Intuitive limits
- Sleeping
- Non-fracture groups

Interfacing with Rigid Bodies

- FEM intended to complement, not replace, rigid bodies
- Have a proof-of-concept integration with rigid bodies
  - Library supports constraints between them
  - Requires coordination of solvers
    - Sequential constraint solve can switch between systems
- Future work
  - Combine with full rigid body solution
  - Switch between FEM and rigid bodies for LOD



Multithreading

- Parallelism
  - Across objects
  - Across collision pairs
  - Across groups of objects in contact (islands)
  - Within an island
- Parallelism within an island solve
  - Multiple constraints affect the same object state
  - Gauss-Seidel must operate on these sequentially
  - Partitioning objects and contacts
  - Analyze independence
  - Schedule with task graph

Threading Approach

- Library uses all async threading
  - Dispatched tasks detect completion of work
  - One of these tasks submits a task to continue
  - Handles all the task synchronization
- High level threading features implemented in the library
  - Parallel for
  - Task graph
- Has a callback interface to support external task systems
  - Mostly just async task submit

Rendering the Deformation

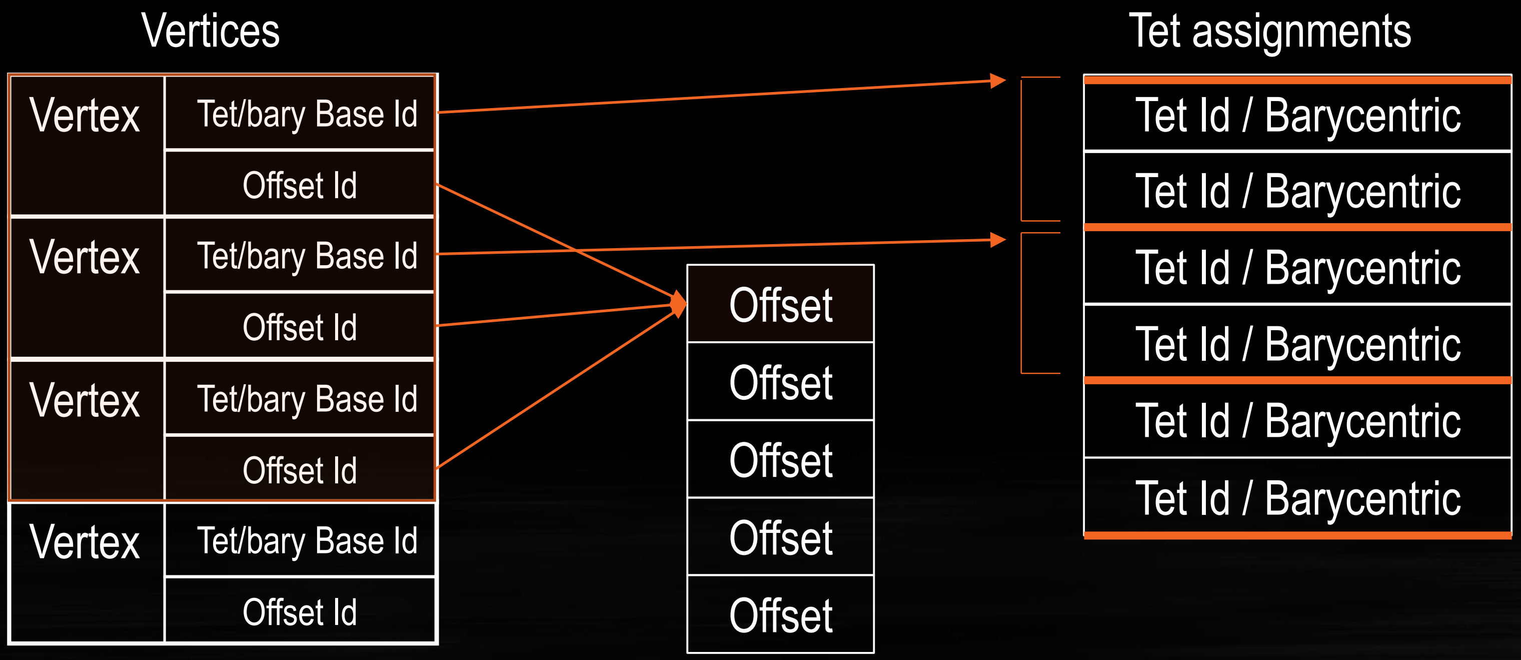
- Render mesh is separate from tetrahedral mesh
  - Allows more visual detail than simulation detail
- Preprocessing finds correspondence between them
  - Attaches each render vertex to closest tetrahedron
  - Finds barycentric coords of vertex in tetrahedron (four weights)
- At run-time vertices driven by tetrahedral mesh and weights
  - Like skinning
  - Can be done in a vertex shader

Handling Fracture

- Render meshes authored in pieces
  - Which break up with fractures in the tetrahedral mesh
- After fracture, may re-assign vertex to another tetrahedron
  - To keep vertices of the same render mesh piece together
  - Uses a map from tet mesh faces to vertices that need re-assignment

Updating the Tet Assignments

- Vertices with same tet assignments are grouped
  - Each has a different base index
  - All point to same offset value, so changing offset updates entire group



Rendering using Unreal Plugin (4.18)

- Found UE4 very flexible for our needs
- Have been able to iterate on it entirely within UE4
- Basic parts
  - Derived UMeshComponent
    - Instance of FEM geometry
    - Associated derived FPrimitiveSceneProxy
      - Buffers used by the rendering thread
  - Derived FVertexFactory
    - For setup of the vertex deformation shader
    - Associated derived FVertexFactoryShaderParameters

Followed Examples from Engine

- These were very helpful
  - ProceduralMeshComponent
  - LocalVertexFactory
- Added custom attributes in FVertexFactory
- Added custom resource parameters in FVertexFactoryShaderParameters
- Added custom resources in FPrimitiveSceneProxy
- Followed LocalVertexFactory.ush but added the vertex deformation
  - GetVertexFactoryIntermediates()
  - VertexFactoryGetWorldPosition()

Scene Proxy Buffer Updates

- CPU buffer updates needed
  - Tetrahedral geometry
  - Tet assignment offsets
- Used Structured Buffers
  - Usage flags:
    - BUF_Dynamic | BUF_ShaderResource
    - CPU write, GPU read
  - Enqueued updates for Render Thread
    - Lock with RLM_WriteOnly

FEM Plugin Design

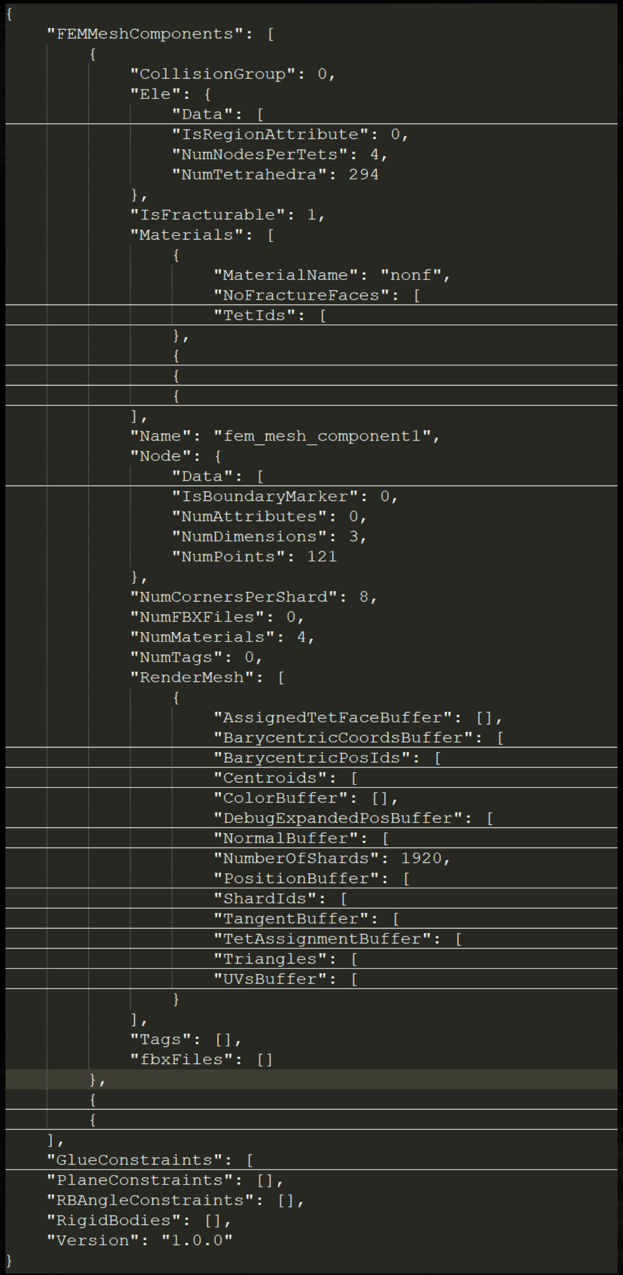
- Developed by Ryan Mayne
- Design philosophy
  - Make it look as much like engine code as possible
    - Tried to find every relevant example in the engine
  - Give maximum control to artists and designers
    - Support everything in blueprints, no C++ required
    - Expose parameters in editor
- Developed along with demo
  - Helped to quickly design scenarios
  - Helped drive library features

Plugin Usage Overview

- Import a .fem file into the engine
- Create a FEMComponent
- Add a FEMScene into the map and set its bounds properly
- Attach a FEMComponent to an actor and set its parameters

FEM Assets

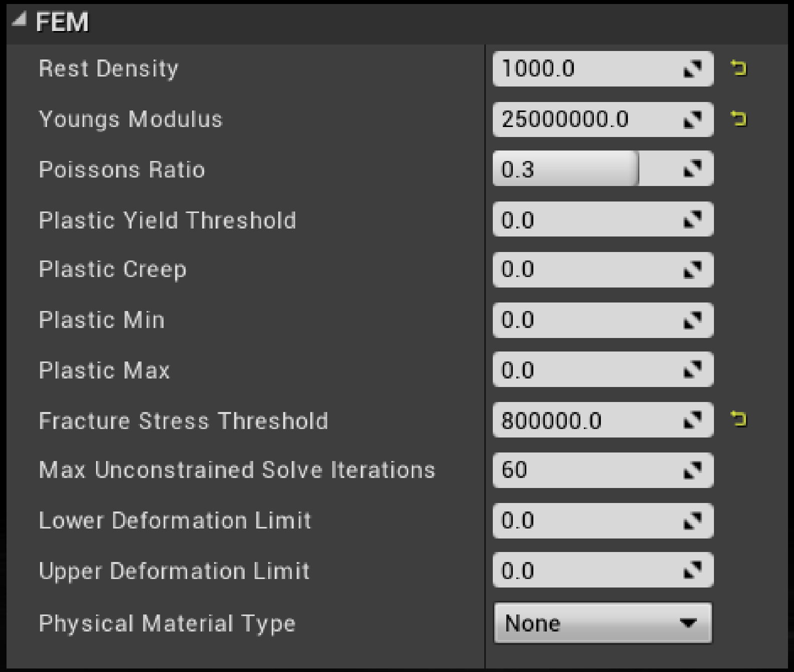
- Developed .fem filetype in the JSON format
    - Create and export with our Houdini tools
- Contains all information for an FEM object
    - Tet mesh geometry
    - Render mesh geometry
    - Tags
    - Rigid bodies
    - Constraints
- Everything imported into Unreal using a custom factory



```
{
    "FEMMeshComponents": [
        {
            "CollisionGroup": 0,
            "Ele": {
                "Data": [
                "IsRegionAttribute": 0,
                "NumNodesPerTets": 4,
                "NumTetrahedra": 294
            },
            "IsFracturable": 1,
            "Materials": [
                {
                    "MaterialName": "nonf",
                    "NoFractureFaces": [
                    "TetIds": [
                },
                {
                {
                {
            ],
            "Name": "fem_mesh_component1",
            "Node": {
                "Data": [
                "IsBoundaryMarker": 0,
                "NumAttributes": 0,
                "NumDimensions": 3,
                "NumPoints": 121
            },
            "NumCornersPerShard": 8,
            "NumFBXFiles": 0,
            "NumMaterials": 4,
            "NumTags": 0,
            "RenderMesh": [
                {
                    "AssignedTetFaceBuffer": [],
                    "BarycentricCoordsBuffer": [
                    "BarycentricPosIds": [
                    "Centroids": [
                    "ColorBuffer": [],
                    "DebugExpandedPosBuffer": [
                    "NormalBuffer": [
                    "NumberOfShards": 1920,
                    "PositionBuffer": [
                    "ShardIds": [
                    "TangentBuffer": [
                    "TetAssignmentBuffer": [
                    "Triangles": [
                    "UVsBuffer": [
                }
            ],
            "Tags": [],
            "fbxFiles": []
        },
        {
        {
    ],
    "GlueConstraints": [
    "PlaneConstraints": [],
    "RBAngleConstraints": [],
    "RigidBodies": [],
    "Version": "1.0.0"
}
```

FEMMesh & TetMeshParameters

- Key was developing our own uassets
  - Enabled serialization, packaging
  - Ability to make objects visible within editor before playing map
- FEMMesh
  - uasset equivalent to .fem source file
- TetMeshParameters
  - Defines the FEM material parameters



FEM

Rest Density	1000.0
Youngs Modulus	25000000.0
Poissons Ratio	0.3
Plastic Yield Threshold	0.0
Plastic Creep	0.0
Plastic Min	0.0
Plastic Max	0.0
Fracture Stress Threshold	800000.0
Max Unconstrained Solve Iterations	60
Lower Deformation Limit	0.0
Upper Deformation Limit	0.0
Physical Material Type	None

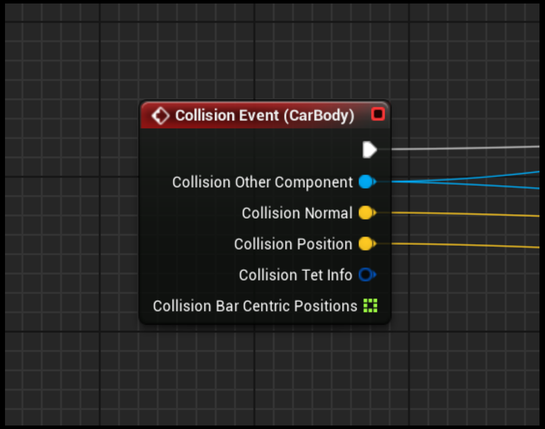
FEM Asset Factories

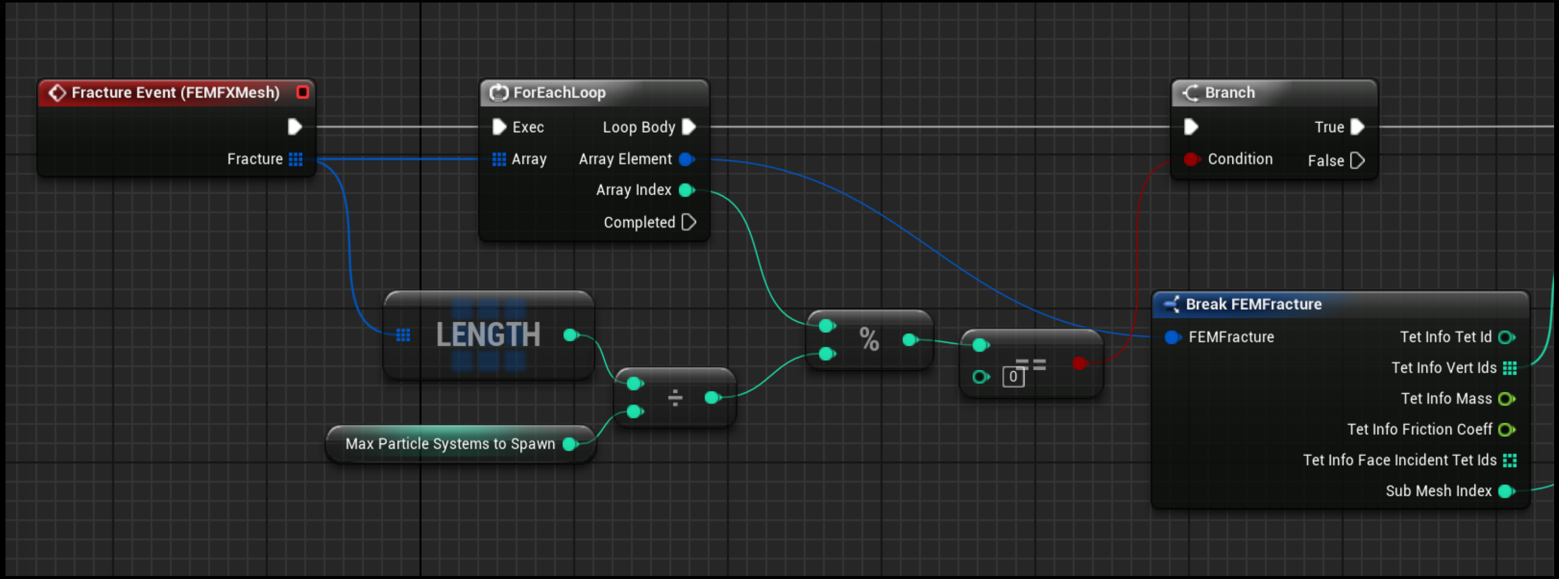
- Each uasset required a factory
- FEMMesh factory
  - Overrides FactoryCreateFile function
  - Loads .fem file using built-in Json module
- TetMeshParameter factory
  - Overrides FactoryCreateNew to create the UFEMTetMeshParameters

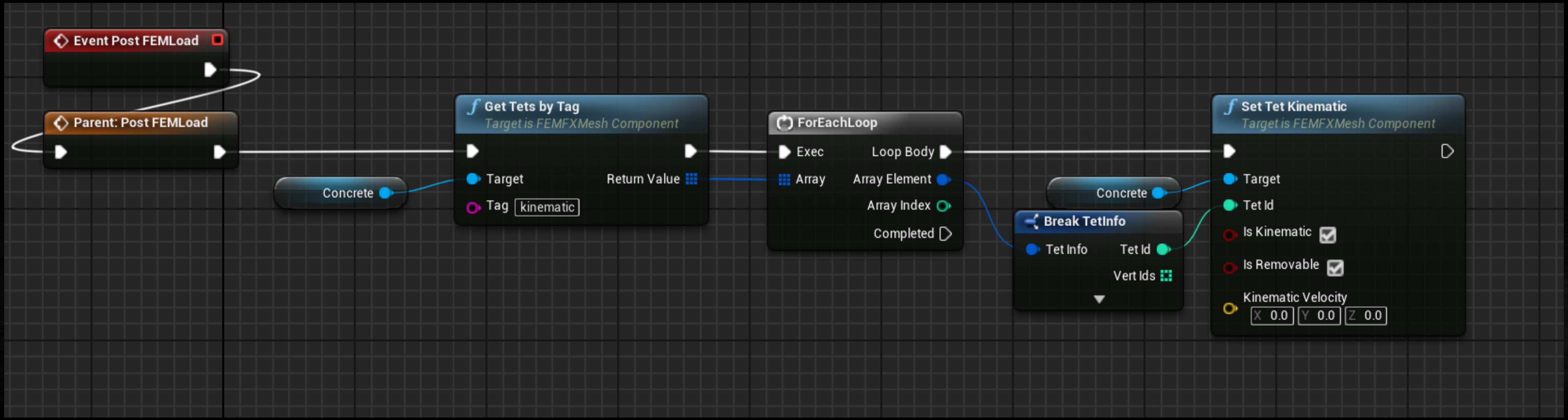
Tags and Events

- Can author tags for each tet
  - Stored in .fem file and imported in UE4
    - E.g., “Kinematic”, “Support Beam”, “Rigid”, “Wood”
  - Allows search through TetMesh for tets with specific tags
    - To apply TetMeshParameters or make immovable for example
  - Powerful feature when combined with UE4 blueprints
- Collision Events and Fracture Events
  - Custom FEM events that can be captured
  - Information exposed by FEM library
  - Really easy to attach gameplay events, audio, and effects

Blueprint Interface Examples

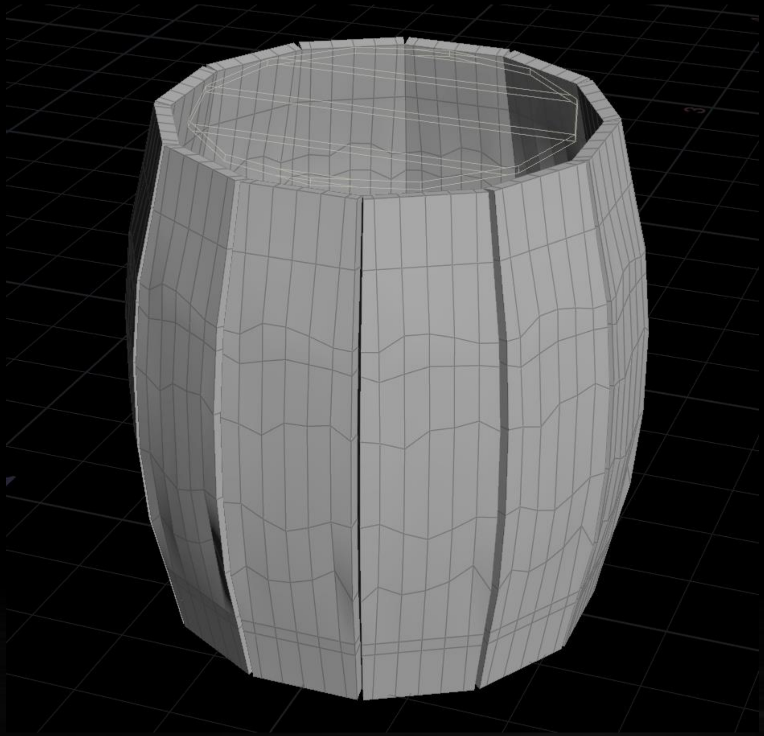

Collision Event (CarBody)
Collision Other Component
Collision Normal
Collision Position
Collision Tet Info
Collision Bar Centric Positions





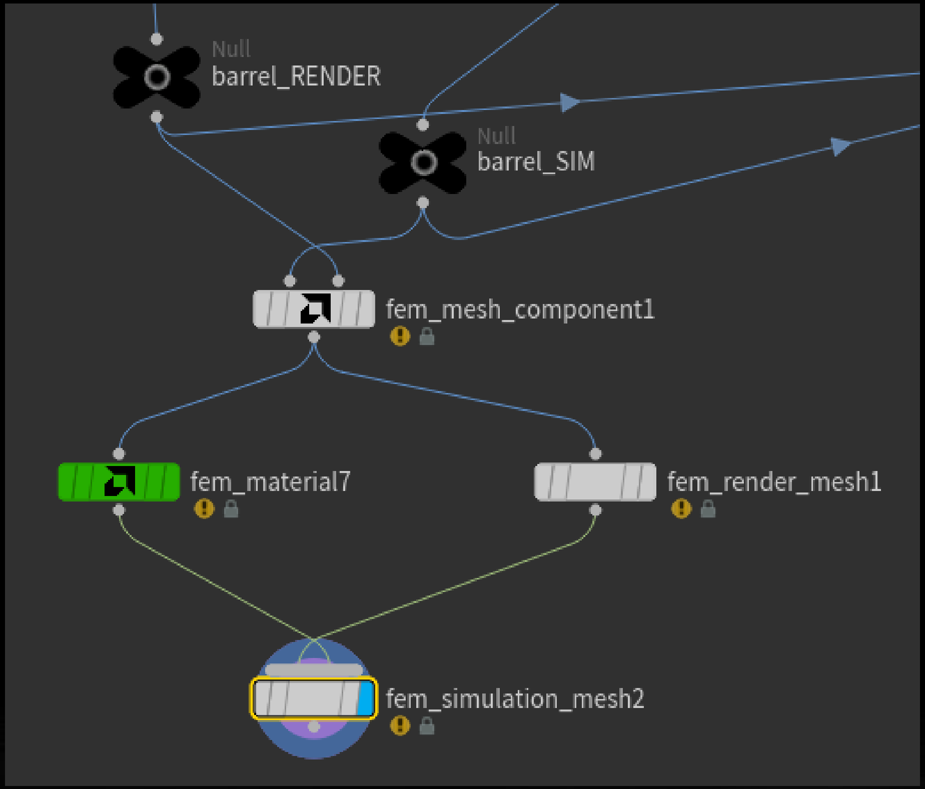
Content Creation

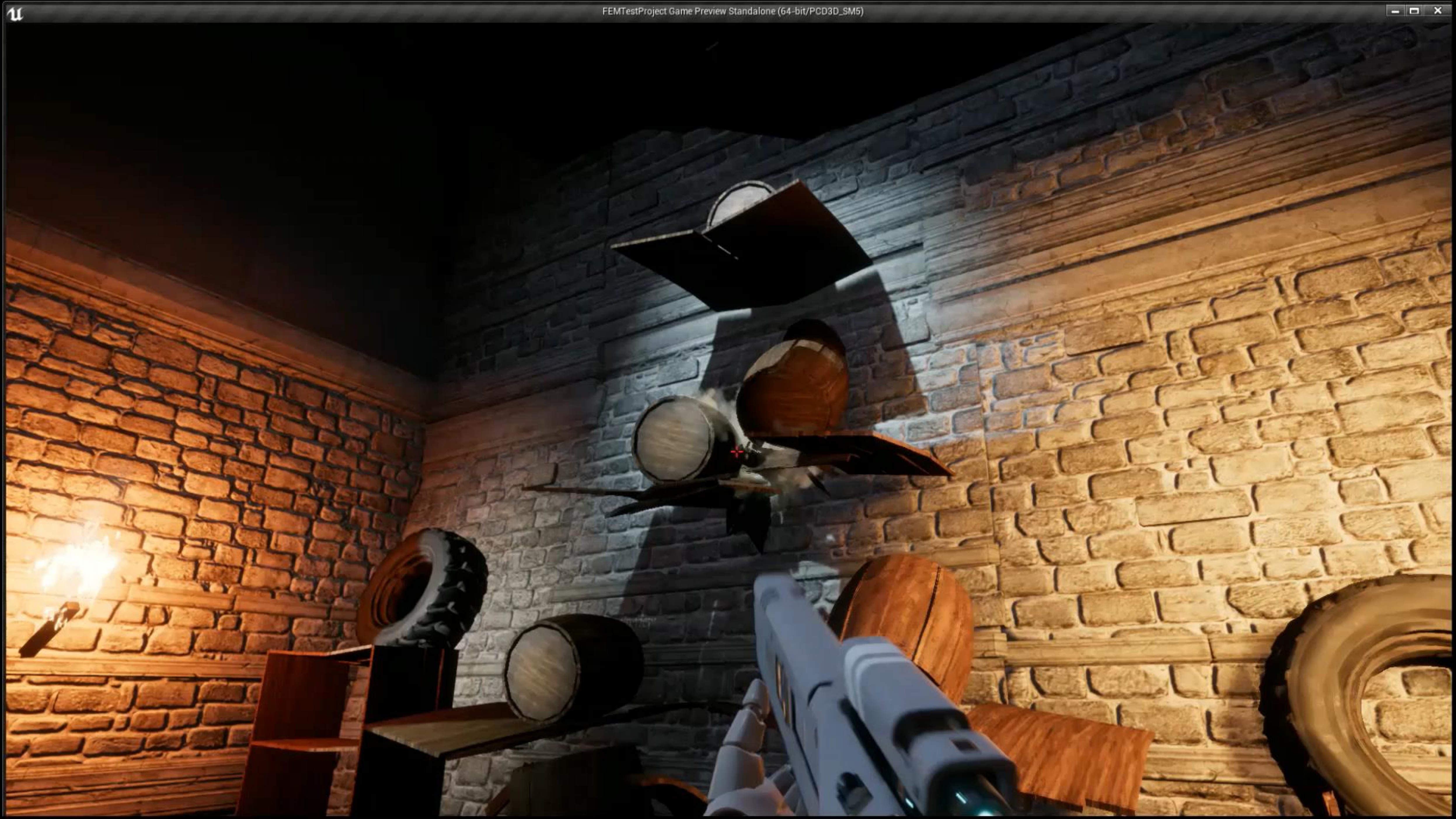
- Houdini-based tools
  - Developed by Joseph Gremlich
  - Creating tetrahedral mesh from input surface
  - Preprocessing render meshes for deformation
  - Select tets to apply properties
    - Materials or tags
    - Marking as non-fracturing group
    - Marking as kinematic
- Artist-friendly workflow
  - Rapid iteration
  - Enabled more complex and interesting examples



Houdini Integration

- Authored several SOP (surface operator) level nodes
  - For geometry and attributes required for plugin
  - Only inputs are simulation surface mesh and render mesh
- Outputs to .fem file format
  - Imported by UE4 plugin





FEMTestProject Game Preview Standalone (64-bit/PCD3D_SM5)

Summary

- Showed our work on FEM material simulation
- Showed how UE4 plugin interfaces served us well
  - Customized rendering
  - Fast artist-driven demo creation

Demo credits

- Chris Woods: FEM content and demo design
- Erasmus Brosdau: Environment design
- Ryan Mayne: Plugin and demo design
- Joseph Gremlich: Tools and visual effects
- Cynthia Anderson: Visual effects
- Tom Perry: Optimization and audio integration
- Josh Kerekes: Audio design

References

- [Baraff and Witkin], “Large Steps in Cloth Simulation” 1998
- [Müller and Gross], “Interactive Virtual Materials” 2004
- [Parker and O’Brien], “Real-Time Deformation and Fracture in a Game Environment” 2009
- [Otaduy et al.], “Implicit Contact Handling for Deformable Objects” 2009
- [Miguel and Otaduy], “Efficient Simulation of Contact Between Rigid and Deformable Objects” 2011
- [Allard et al.], “Volume Contact Constraints at Arbitrary Resolution” 2012
- [Smith and Dodgson], “A topologically robust algorithm for Boolean operations on polyhedral shapes using approximate arithmetic” 2006
- [Cline and Pai], “Post-stabilization for Rigid Body Simulation with Contact and Constraints” 2003
- [Ascher and Boxerman], “On the modified conjugate gradient method in cloth simulation” 2003
- [Irving et al.], “Invertible Finite Elements for Robust Simulation of Large Deformation” 2004
- [Curtis et al.], “Fast Collision Detection for Deformable Models using Representative-Triangles” 2008
- [McAdams et al.], “Computing the Singular Value Decomposition of 3x3 Matrices with Minimal Branching and Elementary Floating Point Operations” 2011

DISCLAIMER AND ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18