Scene Graphs are a data structure for modelling graphical scenes in the computer. They are based on directed acyclic graphs (DAG) where leaf nodes represent geometric shapes and interior nodes represent shape properties, such as color, texture, or position. These properties are inherited along the edges of the graph.

The Scene Graph Caching system proposed in this work compiles the scene graph into a render cache that—when executed—will yield the same output image as rendering the scene graph using the traditional algorithm.

Performance Test Results
The graph on the left shows results from a static scene with approx. 800,000 triangles distributed over a varying number of shape nodes (x-axis). Although the GPU workload is roughly the same for all configurations, using render caches can be up to 4 times as fast for higher node counts. Optimizing the instruction stream brings additional performance gains.

A Render Cache is a small program consisting of GPU commands and their arguments. Executing this program will instruct the graphics hardware to produce an output image. Executing a render cache is very efficient because instruction arguments are already prepared in graphics memory. Like a normal computer program, render caches can be further optimized to run even more efficiently.

Traditionally, a scene graph is rendered by traversing the graph and collecting shape properties along the way. When a leaf node is reached, the shape is drawn with the current set of properties. This algorithm is simple but can become a performance problem for graphs with large numbers of nodes and edges.

The scene graph caching system creates and maintains metadata on the data dependencies between scene graph nodes and instruction arguments. This allows the system to update the render cache incrementally as long as the structure of the scene graph does not change. This way animated scenes containing moving objects can be cached too.

The graph on the right shows normalized frame rates from a scene with a varying percentage of moving objects. Updating the render caches incrementally enables the system to sustain performance gains when the scene contains dynamic content. With optimized (multi-core) cache updates, the speedup is nearly constant for all percentages of objects moving every frame.