Decision Tree Description

The attached image gives an overview of our plans for this semester. The blue nodes represent a stage in the project and other nodes are coloured acording to estimated complexity of the task. The complexity ranges from low (green) to high (red). Colourless nodes are used merely for structure. The work load for the project can be surmised as follows:

- Research 30%
- Modification 30%
- Verification 40%

Live Version

Research

- **Science and Industry**
  - Compare the architecture described in academic litterature with that of open source engines.
- **Engine Map**
  - Explore and map the architectures of open source engines.
- **Unreal vs Godot**
  - Compare an older engine (Unreal) with a new engine (Godot).
- **Crosscutting**
  - Facade Interconnection: Performance/Usability impact of facade pattern in game engine.
  - Inter-Runtime Costs: Performance costs of calls across runtimes.

Modification

- **Physics**
  - Rewrite the physics component using a new technique. This could be functional, parallel or some other relevant approach.
- **C++**
  - Functional Style: Rewrite a C++ component in functional style and compare performance.
  - Modern Style: Rewrite a component in contemporary C++ and compare performance.

- **F# Component**
  - Integrate F# into the engine and allow gameplay code to be written in F#.
  - Exisiting component implemented in F#.
- **Data Flow Component**
  - Use the data flow approach to rewrite a component.
- **Lazy Component: Terrain**
  - Use lazy evaluation strategey to provide on-demand terrain generation.
- **Lazy Component: Numerical**
  - Make lazy numerical evaluation available via a new module.
- **Lazy Component: A.I.**
  - Use machine learning and A.I. techniques with lazy evaluation strategey to progressively improve behaviour.
- **.Net Core: Runtime Switcharoo**
  - Integrate .Net Core in a engine instead of Mono.
- **.Net Core: C# Component**
  - Rewrite a Component in C#.
- **Parallelism: Wrapper Layer**
  - Introduce parallelisme via a layer around an existing component.
- **Parallelism: Parallel Component**
  - Rewrite a component using parallelism techniques available in the current language.

Verification

- **Performance**
  - Microbenchmarks
  - Macrobenchmarks
  - Application benchmarks
  - Memory Analysis
  - GPU benchmakr
- **Usability**
  - Cognitive Dimensions Analysis
  - Instant Data Analysis
  - Discount Usability
  - API Evaluation
  - Git Commit Analysis
    - Analyse commit history of an open source engine and count numbers of “fix, bug” etc.