A0: A Tiny Entity Component System in C++

Course: CPSC 427 - Sep 2023
Due: see course schedule

Note: Although we encourage you to play with different ECS implementations, you will not have to hand in any code for this assignment. Your assignment will be to grade reference implementations on MTA. It both serves to introduce you to the peer grading process and, we believe, teaches the ECS concept more effectively compared to filling code placeholders.

Note that the peer grading is checked and assessed by the TAs. Like assignments, peer grading counts to the final grade. A more precise and detailed grading yields a better grade (see course webpage).

Below, you will find instructions to set up your environment to run the reference solutions, as well as the assignment description for the solutions you will be grading.

1 Introduction

This assignment will prepare you for implementing your own games more efficiently and conveniently by using the entity component system (ECS) pattern that is widely used in game development. It is intended to bring everybody on the same page in terms of programming essentials and sets the foundation for the other assignments and the course project.

2 Installation

This and future projects in the course will use CMake. If CMake is not already installed, you can download and install it from the CMake website: `https://cmake.org/download/`; or use the package manager of your choice on Linux systems.

On Windows, it should be sufficient to open the repository folder (the one containing the CMakeLists.txt) with Visual Studio and hit Build. If you do not see the Build menu option, you may have to install "C++ CMake Tools for Windows". See the Microsoft documentation for details on how to install this workload and on using CMake with Visual Studio: `https://learn.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio#installation`.

On Linux systems, create an empty directory as the build directory, which we assume is named `build`. We recommend using `template/build`. Configure the .gitignore file to ignore any files with `build` in the path, to avoid tracking temporary files. You can configure the project using CMake GUI or the command line. For the GUI, enter the repository folder

(which should contain a CMakeLists.txt file) as Source and the `build` folder as the Build. Then, press configure, and if the configuration if successful, press generate.

For the command line, cd inside the `build` and run:

```
cmake [path_of_assignment_template] -DCMAKE_BUILD_TYPE=[Debug|Release]
```

Now you can build the generated project using make or your favorite IDE (e.g., Visual Studio Code). You require a compiler that supports C++14.

2.1 Running and Debugging

To verify that the installation was successful, you can download one of the submissions you have been assigned to review from MTA. The submission should be a ZIP file. Unzip it using the tool of your choice, and verify that it contains the following files:

```
src/tiny_ecs.cpp
src/tiny_ecs.hpp
ecs_demo.cpp
```

The program should print a few lines to the terminal and then exit. Set a breakpoint in the `main()` function in `ecs_demo.cpp` and run in debug mode until that point to verify that you can use your IDE's debugging capabilities. Make yourself familiar with the inspection of local and global variables and stepping through instructions in debug mode.

3 Required Work

This section contains the instructions for the assignments that you will be grading on MTA. We encourage you to think about how you would implement the following features of an ECS system, and experiment with different implementations yourself, but once again, you do not need to hand in any code for the following tasks.

3.1 Entity Component System (ECS) basics

The ECS pattern stores the entire game state into components that are associated to entities. Each entity uniquely identifies an object, such as a fish or turtle in the game; whereas the components equip entities with properties, such as name and movement capabilities. This compositional scheme is an alternative to inheritance, which, among other shortcomings, suffers from the multiple-inhertance problem, also called the diamond problem.

Your task is to implement the `Sparse Array` or the `Dense Array + Map` approaches introduced in the lecture. Implement a `Registry` class that supports the following functionality:

Task 1: (5 %) Creating a new entity with a unique ID.

Task 2: (15 %) Adding a new component for a given entity.

Task 3: (10 %) Checking whether an entity has a certain component.

Task 4: (10 %) Removing a component from a given entity.

Task 5: (10 %) Getting the component associated with a given entity.

To validate proper functioning, write the following tests into `ecs_demo.cpp`. No graphical user output is required, simply print states to the terminal.

Task 6: (10 %) Create a Salmon entity that can swim with speed 3.

Task 7: (10 %) Create a Chicken entity that can swim, fly and walk.

Task 8: (10 %) Add the ability to fly to the existing salmon and remove the ability to fly from the chicken, as the salmon can jump up the river and most chickens can't fly.

Task 9: (10 %) Change the swim speed of the existing salmon to 5 more than its previous speed.

Task 10: (10 %) Equip the salmon and chicken entities with x and y positions. (Hint: you should create and add one or more new components to represent an entity's x and y position.) Then, explain whether the Registry stores the x and y position information in an Array of Structs (AoS) layout or a Struct of Arrays (SoA) layout.

Make sure that your changes are reflected in the output. In particular, when modifying properties of components stored in the ECS registry, make sure to work on a reference to the component or create a pointer to it. It is a common mistake to work with a copy of the object, which does not change the original properties stored in the ECS registry.

3.2 Optimization

C++ is a statically typed language, which brings many performance advantages ($\sim$ 10-times faster than the interpreted python code and $\sim$ 2-times faster than just-in-time compiled Julia language). However, this leads to complications when working with dynamic objects, such as defining a function that can work on multiple argument types. See the following article for a detailed introduction on using templates `http://www.cplusplus.com/doc/oldtutorial/templates/` and attend the course tutorials to get an interactive introduction.

Bonus Task 1: (1 %) Use template classes or template functions to collect components of an arbitrary type and to associate each component with an entity without having to duplicate code.

Bonus Task 2: (1 %) Use move operations (`std::move`) to avoid unnecessary copies and improve performance.

Bonus Task 3: (1 %) Add an `emplace` function using parameter packs (a C++11 feature). The `emplace` function should accept zero or more arguments, construct a new component using the provided arguments, and insert it into the ECS registry.

4 Grading

For full points the solution must be correct, concise, and reasonably documented with comments.

For peer grading, check whether the features in the grading rubric on MTA are fulfilled and deduct points accordingly.

5 Handin Instructions

1. Zip all code and CMake files as present in the template source code but exclude any executables and compiled files. Upload the zip file via MTA. Double check that you excluded all generated files, such as /build, .vs, /out! These would consume a lot of space on our server.

2. In addition, create a README.md file (Markdown language as used on GitHub) that includes any information you would like to pass on to the marker.

Note, do not publish your solution (OR the solutions given to you for grading) on GitHub or any other place. Neither during the course nor after; both are considered cheating.