ECE 3574: Applied Software Design

More Design Patterns

Spring 2017

Today we will discuss the design patterns Factory, State, and Model-View.

- Factory Pattern
- State Pattern
- Model-View (and Model-View-Controller) Pattern

Design Pattern: Factories

When using dynamic polymorphism it is common to have many types that derive from a common base with the subtype specified at runtime.

The Abstract Factory pattern is a class that builds subtypes based on a description, usually derived from user input, and returns a base pointer to the constructed object. This collects switch-based object construction code into one place.

See example code.

Note, this works best when using the object does not require casting (as in good polymorphic design).

Remember destructors for base classes should be virtual

The example I showed used stack allocated objects inside the derived class.
If you have to *manage resources* in the derived class, make sure the base class has a virtual destructor

Design Pattern: State (as in State Machine)

The state pattern uses a private pointer to a state object to encapsulate behavior based on the state an object is in, with the ability to transition.
See example code.
This is usefull whenever you need to cleanly code a complex state machine.

Design Pattern: State (as in State Machine)

The state pattern uses a private pointer to a state object to encapsulate behavior based on the state an object is in, with the ability to transition.
See example code.
This is usefull whenever you need to cleanly code a complex state machine.

Model-View Pattern

The model-view pattarn separates data (the model) from the code used to present it (the view).
Communication happens through a well-defined interface.
Thus any object that conforms to the interface can be viewed without custom code.
See example1 and example2 code.

Model-View-Controller Pattern

For interactive applications (e.g. GUI) it is common to introduce a third object called the controller that mediates between user actions, the view, and the model.
Typically in Qt the models, views, and controllers communicate using a mixture of events and the signal/slot mechanism.
See example3 code.

Next Actions and Reminders

- Read Chapter 26 of Operating Systems: Three Easy Pieces
- Project 2 beta is due 3/28 at 8 am.