Lesson 11: Adding New Variables to a Level Object

3D Game Programming With C++

Digital Media Academy (Summer 2011)

Written by: Andrew Uzilov (andrew.uzilov@gmail.com)
Feel free to contact me with any questions.

In the previous lesson, we used the variables (more specifically, objects) named `player` and `camera` that were already declared and defined for us. But what if we want to add new variables that we want to use in both `init()` and `logic()` functions? Where do we declare them? Where do we define them? This lesson will answer these questions.

CODE ORGANIZATION

In any moderately complex C++ project, you will find that there are two types of files: ones that end with the `.cpp` (sometimes `.cxx` or `.c++`) suffix, and others that end with the `.h` suffix. They come in pairs: two files with the same name, but different suffix, like this:

```
LevelOneState.cpp
LevelOneState.h
```

Why is this? Simply put, this is a convention for organizing object-oriented code. **Declarations** go into the `.h` file. **Definitions**, which is where values are actually assigned to variables, go into the `.cpp` file. The same applies to functions and classes. Functions and classes are declared in `.h` files and defined in `.cpp` files.

Whenever you use `#include` to include code from another library, you usually give it the name of a `.h` file, not the corresponding `.cpp` file. This way, the compiler knows all the variable, function, and class names, but doesn't have to recompile all the code that defines what they are.

ADDING A NEW VARIABLE

Let's add a new `double` variable called `speed` to a `LevelOneState` object. Every time we want to add a variable to a level, we need to do two things:

1) Declare the variable in the `.h` file
2) Define it in the corresponding `.cpp` file in the `init()` function

In order to be usable in both `init()` and `logic()` functions, we have to put the variable declaration inside the `LevelOneState` **class declaration**. The new variable should go into here:

```
class LevelOneState: public GameState {
private:
	/* Camera stuff.
	 */
	// Handle to the camera.
	NodePath camera;

	/* Player stuff.
	 */
	// Handle to the player.
	NodePath player;

	/* Player stats.
	 * We'll need this someday.
	 */
	PlayerStats *playerStats;

	int speed; // <-- NEW VARIABLE DECLARATION

public:
	LevelOneState();
	virtual ~LevelOneState();
	void logic (void);
	void init (PlayerStats*);
	void deinit (void);
};
```

Now, we have to assign an initial value to it (define it). This can be done anywhere in the `init()` function:

```
void LevelOneState::init (PlayerStats *ps) {
	/* other stuff */
	speed = 100;
	/* other stuff */
}
```

Now we can use this variable in the `logic()` function! We can even change its value in `logic()` so that we can model acceleration.

Exercise 11: A random walk

Make the panda go on a random walk. The panda should slowly walk forward for exactly 50 frames, then turn for 25 frames, then walk again, then turn again, and so on.

The turning speed and direction should be picked at random (look back at Exercise 5.3 for how to do randomness).

Hint: Create a new variable that counts the frame number and initialize it to 0. On every call to `logic()`, increment the variable by 1. If you are on frame 0 through 49, you should be walking forward. If you are on frame 50 through 74, you should be turning. If you hit frame 75, reset the counter to 0.

Hint: You will need to create another variable for storing your "heading change per frame" after you randomly choose it, so you can keep turning at the same speed for all 25 frames during your turning step.