Design and Design Patterns

Design Patterns

Design Patterns use OO-principles to solve common problems.

Singletons: The OO answer to global variables.

Why not use globals?

A. They make code hard to understand.

B. They make code hard to debug.

C. They make code hard to modify.

Why not use globals?

D. Profs O'Neill and Kuenning with haunt your dreams if you do.

Answer

All of the above.

Singleton Pattern

- Problem: Ensure a class has only one instance and provide a global point of access to that instance.

Singleton Class

```
class Singleton
{
public:
        static Singleton* Instance();
private:
        static Singleton* theSingletonInstance;
        Singleton() {};
        ~Singleton() {};
        Singleton(const Singleton& toCopy) {};
        Singleton& operator=(const Singleton& toCopy) {};

};

Singleton::Singleton* theSingletonInstance = NULL;
```

Instance Implementation

```
Singleton* Instance()
{
        if (theSingletonInstance == NULL)
                theSingletonInstance = new Singleton;
        return theSingletonInstance;
}
```

Access

```
Singleton* ptrTheSingleton = Singleton::Instance;
```

Example

```
class Ball
{
public:
        static Ball* theBall();
private:
        Sphere   theSphere;
        Ball() {};
        ~Ball() {};
};

Ball::Ball* theBall = NULL;
```

new problem

I want a 2D graphics library that supports the following functions for triangles:

- set color to r,g,b
- translate vertices by dx, dy
- rotate α degrees about the origin
- draw

help

I have a 3D graphics library that has a triangle class with the following interface

- triangle()
- triangle(v1x, v1y, v1z, v2x, v2y, v2z, v3x, v3y, v3z)
- ~triangle()
- set color(r, g, b)
- rotate(vector, angle)
- translate(dx, dy, dz)
- scale(sx, sy, sz)
- draw()
- flip(planeA, planeB, planeC, planeD)
- texture(textureMap)
- standardize()

exercise

Design a 2D triangle class that uses the 3D class to do the work!

façade

- Scenario — You need to use a subset of a complex system or you need to interact with the system in a particular way.
- Problem — You want to simplify how the complex system is used to fit your application.

Problem

I am building a physics engine that performs collision detection between a sphere and some triangles. My physics engine class contains the following method:

cCollision cPhysicsEngine::detectCollision(cPath p, cTriangles t)

Later I plan to implement a faster but less robust algorithm that could be used on systems with slow processors.

Come up with a design that will allow for future variations in my collision detection algorithm.

Strategy Design Pattern