The Pop Framework

CS134

Chris Pollett

Sep.1, 2004
Outline

• OO Simulations
• The Pop Source Code
• Essential Pop Classes
• UML Class Diagrams
• Using the Pop Framework
OO Simulations

• Games often simulate some aspect of the real world.

• A simulation is a model of real world phenomena that is run on a computer (or any other device other than on the world itself)

• Some examples of things we can simulate: dynamics of motion, motion of group of objects such as rockets, birds, sports balls, etc. Growth of cities, spread of diseases, etc.
OO and Simulations

- OO Design lends itself to simulations. Most simulations break into a variety of objects that are being simulated and these objects called modelled using classes.

- We can wrap into a class definition the properties of an object and defining member functions we can say how the object behaves according to various inputs. This "Wrapping" is called encapsulation.
Yet More OO Stuff

• OO also allows one to give the objects in the simulation a uniform behavior. All instances of an object can behave according to some behavior function. If want new objects which are the same but use a different behavior function we can use inheritance or strategy patterns.

• Example: could override Person to get PersonOnEarth or PersonOnMars where the physics of motion might be different
The Pop Source Code

• Consists of the following groups of file:
  – MFC files: childfrm, mainfrm, pop, PopDoc, popview, stdafx
  – Game files: game, gameairhockey, gameballworld, gamedambuilder, gamepicknpop, GameSpacewar, gamestub, gamestub3d
  – Critter files: biota, critter, critterarmed, critterwall, critterviewer
More Source Files

- Sprite files: sprite, spritebubble, spritepolygon, spritemultiIcon.
- Physics files: VectorTransformation, realbox, force
- Utility files: controller, listener, metric, Randomizer, timer
- Graphics files: graphics, graphicsMFC, graphicsOpenGL, memoryydc, RealPixelConverter, texture, glshapes
- Dialog files: SpeedDialog
- Parameter and Resource files
Pop Build Involves

-->h-->cpp-->obj\
-->vcproj --> sln -->exe
    -->bmp\ /
    -->cur---->rc-->res/
-->wav/
-->ico/
Essential Pop Classes

Some of the classes + basic UML

- cCritter
- cSprite
- cGame
- cCritter
Main Pop Classes

CPopApp

CPerformanceTimer

CPopDoc

CPopView

cGame

cCritter

cGraphics

cSprite

cListener

cForce
One of the most important classes is `cCritter`. It is used for the basic objects in the Pop world. Some subclasses are: `cCritterArmed`, `cCritterBullet`, `cCritterArmedPlayer`, `cCritterArmedRobot`, `cCritterWall`, `cCritterViewer`, etc.
cSprite

- cCritter delegates to cSprite the responsibility of how a critter actually looks
- Can change appearance of a critter without have to create a new class.
- cSprite has a draw method to do drawing
- cSprite has subclasses cPolygon, cSpriteIcon, cSpriteDirectional, cSpriteLoop, cSpriteCircle, cSpriteBubble, etc
cGame

- Initializes all the creatures in the world
- Keeps track of each critter's status and status of game
- Has a function `step(dt)` that simulates `dt` amount of time

Everything is stored as a real number (i.e., `dt`) to avoid resolution dependence. `cVector` used to specify points in world. There is a related `cMatrix`. 
cForce

- A critter’s behavior can be influenced by any number of force objects.
- Some example subclasses are cForceGravity, cForceDrag, cForceObjectSpringRod, cForceObjectSeek, cForceEvadeBullet.
cListener

• At each update a critter’s cListener object is given access to the current mouse and key state and is allowed to change the critter’s motion or state.

• Some subclasses include cListenerArrow, cListenerScooter, cListenerCursor, etc.
CPopApp

- This is the main application class. It is an MFC subclass
- OnIdle has been overwritten to drive the animation.
- Calls cPerformanceTimer to find the time since last update which is used eventually to call step(dt)
CPopDoc

• Document that holds the data associated with your windows and game you are running
CPopView

• Is a view that controls how the data is displayed in an onscreen window. Also does initial processing on user input.
• Delegates actual drawing to a cGraphics object which is a bridge to an underlying graphics system like GDI or OpenGL.

Subclasses of cGraphics include cGraphicsMFC and cGraphicsOpenGL
UML Class Diagrams

• Have seen has-a relationships already. To draw subclasses can use:

```
  cGame
  |    
  v    
  cGameStub

  cCritter
  |    
  v    
  cCritterArmed
    |    
    v    
    cCritterStubPlayer
    |    
    v    
    cCritterStubProp

  cCritter
  |    
  v    
  cCritterStubRival
```
Associations

• Class has an instance of another object or has a function which returns the other object.

  ![Diagram showing association between cGame and cCritter]

• Can add arrows to indicate has a way to navigate to an object
Using the Pop Framework

• Want to subclass the class cGame. (In homework we take cGameStub and modify it)
• Edit CPopDoc so that it sets your game as the default game:
  setGameClass(RUNTIME_CLASS(cGameMyGame));
• Probably will want to modify resource files cCritterStubPlayer and cCritterStubRival and cCritterStubProp. For example, setMoveBox function of critter controls region critter can move in