Listeners

CS134

Chris Pollett

Oct 13, 2004.

Outline

- cController
- From keypress to critter
- Listeners
- Shooting with Listeners
- Viewer listeners
- Listeners initializing critters

cController

- In MFC, void CView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) is triggered whenever a key is pressed.
  - Here nFlags is a set of bitflags saying if Ctrl, Alt, and/or Shift are being pressed
  - nRepCnt is supposed to hold the number of keypresses caused by holding that key down
- In Pop, the class cController is used to hold the state of the keyboard and mouse.
- It has a useful accessor functions to get this info.
- Possible keys are represented as in Windows by integer keycodes (#defines VK_???. Ex: VK_A, VK_LEFT, etc)

More on cController

- cController maintains an unsigned integer for each key representing the current keystate.
- Bit-flags in this integer indicate if Shift or CTRL also pressed.
- Flags also say if a key has been down for more than one cGame::step call.

cController Class (partial defn)

```
class cController : public CObject
{
    protected:
        UINT _keystate[VKKEYCOUNT];
        Real  _keystateage[VKKEYCOUNT];

    public:
        cController();
        virtual void update(Real dt);
        BOOL keyon(int vkcode);
        BOOL keyonplain(int vkcode);
        BOOL keyoncontrol(int vkcode); …
};
```

From keypress to critter

When you press a key:

- An OnKeyDown event goes to the active CPopView
- CPopView::OnKeyDown calls cGame::onKeyDown
- The cGame object stores the key information in its *_pcontroller
- The cGame::step method calls cCritter::feellistener of the player
- The Player's critter calls _plistener->listen(dt, this) where _plistener is a pointer to a cListener
- listen uses pcritter->pgame()->pcontroller() to get at _pcontroller to see which keys are currently being pressed
- Depending on this, cCritter::setAcceleration and similar methods are called

Remarks

- Notice any critter can access pcritter->pgame()->pcontroller()
- Can use this for multiperson games

Listeners

- As we already said:

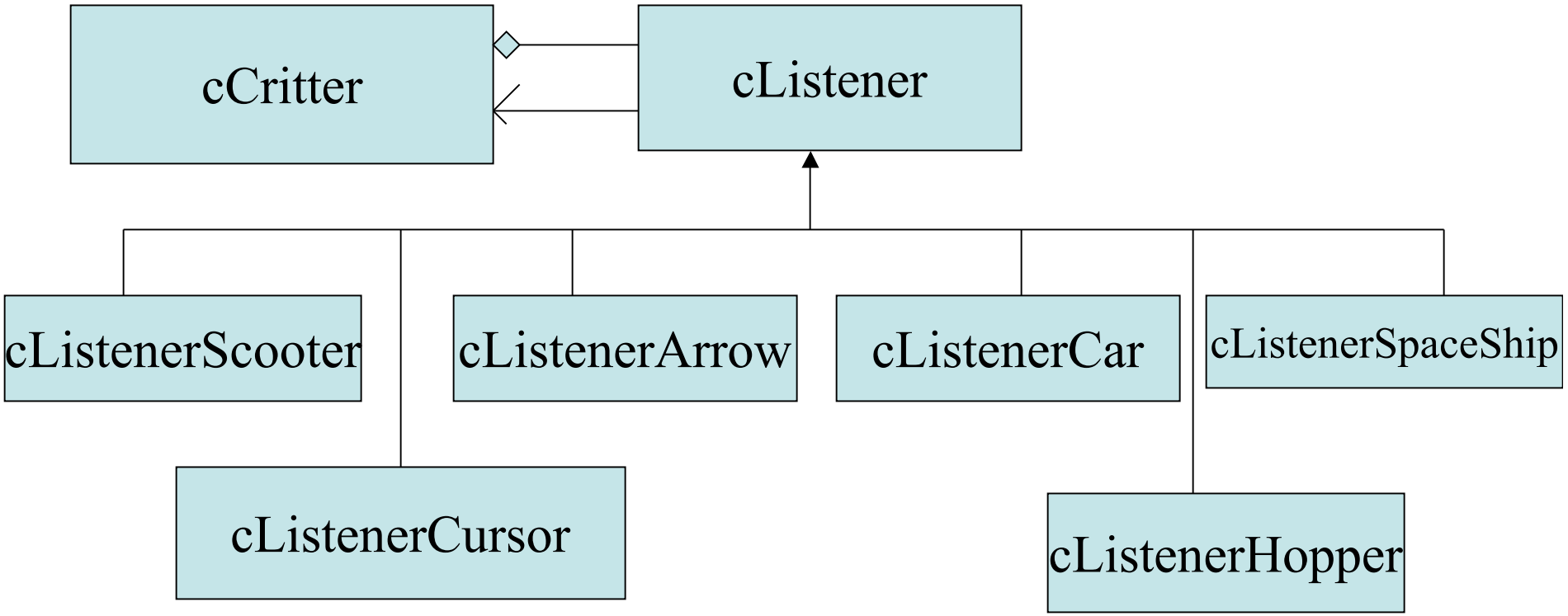
```
void cCritter::feellistener(Real dt)
{
    _plistener->listen(dt, this);
}
```

- We pass this so can adjust fields of cCritter
- Also, pass this so cListener can navigate to cGame
- The dt is passed so adjustments to critters position can depend on how much time has passed.

How feellistener is called

- feellistener is called within cGame::step.
- cGame::step generates calls feellistener(), move(), update(), feelistener(), move(), update()….
- Notice update() will call feelforce() so feellistener typically adds in more acceleration to those just added
- These accelerations are used to calculate the movement done in move()

UML for different kinds of Listeners



Example Listener

```
void cListenerArrow::listen(Real dt, cCritter *pcritter)
{
        cController *pcontroller = pcritter->pgame->pcontroller();
        pcritter->setAcceleration(cVector::ZEROVECTOR);
        if( !pcontroller->keyonplain(VK_LEFT) &&
          !pcontroller->keyonplain(VK_RIGHT) &&
          !pcontroller->keyonplain(VK_DOWN) &&
          !pcontroller->keyonplain(VK_UP) &&
          !pcontroller->keyonplain(VK_PAGEDOWN) && //used in 3D for z direction
          !pcontroller->keyonplain(VK_PAGEUP) &&) //used in 3D
        {
             pcritter->setVelocity(cVector::ZEROVECTOR);
             return;
        }
        if(pcontroller->keyoneplain(VK_LEFT))
             pcritter->setVelocity(-pcritter->maxspeed()*cVector::XAXIS);
        …..
      }
```

More On Example

- End of function adjusts attitude matrix if motion lock on.

cListenerScooter

- Also directly sets the critter's velocity.
- So also sets acceleration vector to zero before changing velocity.
- In scooter:
  - Up key sets the critter's velocity to its maxspeed in the direction of the tangent. (note: if stop pressing key velocity set to zero)
  - The down key sets the critter's veolcity to maxspeed in the opposite direction
  - The left and right arrows cause the critter to yaw. That is, the tangent is rotated about the binormal.
  - Page-up and page-down cause the critter to pitch. That is, the tangent is rotated around the normal.
  - Finally, Home and End keys 'roll' the critter by rotating its normal around the tangent
  - Critter's direction of looking also visibly updated.

More on listeners

- To make things more responsive, it is useful to have two turn speed's
- Some listeners don't act directly on velocity but use acceleration instead: cListenerSpaceship, and cListenerCar both add and subtract from the acceleration.
  - Spaceship adds in the direction currently pointing
  - Car adds in the direction of current motion.
- cListenerCursor sets acceleration to zero, then sets velocity to what is needed to move critter to current mouse location in dt time.

Shooting with Listeners

- For many games, it is useful to have critters that can shoot/eject objects.
- Suppose one wants to allow shooting to be done by hitting the space bar or pushing the left mouse button.
- Shooting critter's are typically derived from cCritterArmedPlayer which can handle this kind of shooting via the code in its feellistener method. The state of shooting or not is toggled via a _bshooting variable.
- Reason this code is done in feellistener is so that code does not have to be executed by listening critters that don't shoot.

Shooting feellistener code

```
void cCritterArmedPlayer::feellistener(Real dt)
{
    cCritter::feellistener(dt);
    _bshooting = (pgame()->keystate(VK_SPACE) ==
       cController::KEYON);
    if(pgame()->keystate(VK_LBUTTON) ==
       cController::KEYON)
    {
        _bshooting = TRUE;
        aimAt(pgame()->cursorpos());
    }
}
```

Viewer listeners

- Each CPopView has a cCritterViewer *_pviewpointcritter that is used to set the projection matrix and view matrix inside the CPopView::OnDraw call.
- A view shows the game world as seen from its _pviewpointcritter.
- The programmer can change the appearance of the view by moving or rotating the _pviewpointcritter, and setting the zoom as discussed last day.
- In order to let the game player change the viewpoint can add a listener to this critter.
  - pviewpointcritter->setListener(new cListenerViewerOrtho());
- cListenerViewerOrtho is one of three special listeners in Pop just for viewers.

More Viewer Listeners

- cListenerOrtho is used for 2D-worlds. Reacts to Ctrl + arrow keys. Moves the _pviewpointcritter back and forth parallel to the XY-plane. Ins and Del generate in out zoom calls.
- cListenerViewerFly and cListenerViewerRide are always used as the _pviewpointcritter listener for 3D-worlds. In fly mode, Ctrl+arrow combinations move the viewpoint critter along its tangent, normal, and binormal directions. Ctrl+Shift+arrow rotates the _pviewpointcritter along these axes. Ins and Del zooms in/out.
- cListenerViewerRide is used to let the viewer ride upon the games pplayer ar a fixed cVector _offset.

Listeners initializing critters

- The cListener class has a virtual void install(cCritter *pcritter) method.
- Recall attach a listener to a critter using cCritter's setListener method(cListener *plistener).
- setListener calls plistener->install(this) to give the listener a chance to make adjustment to the critter before it starts to be used.