Shooters and bullets

CS134

Chris Pollett

Oct 18, 2004.

Outline

- High-level shooter design
- cCritterArmed
- cCritterBullet
- *damage* and *draw*
- Armed players and armed robots
- cCritterArmed/cCritterBullet association

High-level design of shooters

- A class cCritterArmed is used to encapsulate shooting behavior
  - Has new methods aimAt and shoot
  - Overrides draw and update
- cCritterArmedPlayer and cCritterArmedRobot inherit from it.
- A cCritterArmedRobot and a critter pointer _ptarget that it tries to shoot at.
- Have a class cCritterBullet for what is shot.
  - Has new methods initialize and target
  - Overrides update, collide and collidesWith

cCritterArmed

- cCritterArmed::update is responsible for whether shoot is called. It checks:
  - if the critter's _armed flag is on.
  - if the critter's _bshooting flag is on (so critter is currently shooting).
- For robots/rivals the _bshooting flag is continually on and _waitshoot is used to control time between shots

Code for update

```
void cCritterArmed::update(CPopview *pactiveview)
{
	cCritter::update(pactiveview);
	if(_aimtoattitudelock)
		setAimVector(attitudeTangle());
	if(_armed && _bshooting && (_age - _ageshoot >_waitshoot))
	{
		shoot();
		_ageshoot = _age;
	}
}
```

More on cCritterArmed

- For the player class, to make the direction of the gun visible, draw is overriden to draw a line segment under cCritterArmed's sprite
- There is a CRuntimeClass *_pbulletclass variable to keep track of what kind of bullets to use.
  - So don't need to override shoot()
- shoot() does the following:
  - if more than _maxbullets active, deletes oldest
  - creates a pbullet with pbulletclass->CreateObject()
  - calls pbullet->initialize(this) to set up bullet.

cCritterBullet

Has a no-argument constructor which:

- sets the bullet's _collidepriority to cCollide::CP_BULLET. (higher than normal critter)
- sets _usefixedlifetime to TRUE and sets duration of lifetime to cCritterBullet::FIXEDLIFETIME (3sec's)
- makes a yellow isosceles triangle the default sprite
- sets the bullet's speed, _maxspeed, and _hitstrength

cCritterBullet::initialize

- matches the bullet's attitude to the shooter's
- positions the bullet at the tip of the shooter's gun
- sets the direction of the bullet's velocity to match shooter's _aimvector. Speed comes from the constructor. (can view speed as muzzle velocity)
- attaches a copy of the shooter's physics forces to the bullet
- copies the shooter's _movebox to the bullet
- gives the bullet the same _ptarget as the shooter

cBullet::update

```
void cCritterBullet(CPopView *pactiveview)
{
    cCritter::update(pactiveview);
    if(_outcode && _dieatedges) //die when close to edge of
    //world. set _dieatedges false if want bullet's to bounce
    {
        delete_me();
        return;
    }
}
```

cCritterBullet::collide

- collide is where bullet do damage

```
BOOL cCritterBullet::collide(cCritter *pcritter)
{
    if(isTarget(pcritter)
    {
        if(!touch(pcritter))
            return false;
        int hitscore = pcritter->damage(_hitstrength);
        delete_me();
        if(_pshooter) _pshooter->addScore(histscore);
        return TRUE;
    }
    else return cCritter::collide(pcritter);
}
```

cCritterBulletSilver

- Unlike other bullets, silver bullets override isTarget to target only one critter rather than one kind of critter:

```
BOOL cCritterBulletSilver::isTarget(cCritter* pcritter)
{
    return pcritter == _ptarget;
}
```

- _collidepriority is slightly lower than normal bullets -- allows one to shoot at these kind of bullets in Spacewar

damage and *draw*

- The cCritter method damage looks like (might want to override to play a sound):

```
int cCritter::damage(int hitstrength)
{
    if(_shieldflag || recentlyDamaged())
   //recentlyDamaged require a safe amount of
  // time to pass before can be damage again
      return 0;
    _lasthit_age = _age;
    _health -= hitstrength; //health usual starts 1 so this can kill
    if(_health <= 0){_health =0; die(); return _value;}
    return 0;
}
```

Useful to indicate critter temporarily can't be damaged so override draw

How draw indicates recently damaged

```
void cCritter::draw(cGraphics *pgraphics, int
   drawflags)
{
    if(recentlyDamaged())
    {
        drawflags |= CPopView::DF_WIREFRAME;
        //draw in wireframe if just damaged
    }
    //more code
}
```

Armed players

- player shoots when spacebar or left mouse clicked. This is done by overriding feellistener(dt):

```
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());
    }
}
```

More on armed player's

- shoot() in this case adds player speed of motion (does it only if two are going same direction)
- Constructor calls:
  - setAttitudeMotionLock(FALSE) so player can move direction independently of motion
  - sets sprite to be a red isosceles triangle
  - overrides damage to play a sound
  - overrides draw to draw a circle around the player
- Class has a _sensitive field used by collide to cause damage to be called if touch another critter

Armed Robots

- Robot's _bshooting is always true.
- _waitshoot is used to say delay between shots
- Can set with setWaitShoot(Real waitshoot)
- To avoid shooting in synchrony this method adds a little randomness.

cCritterArmed/cCritterBullet association

- Bullets have a *_pshooter field so that:
  - A bullet doesn't shoot its shooter
  - When the bullet damages something, points can be awarded to the player
  - When a bullet dies it can notify its critter
- Armed critters have an array of bullets shot so that
  - If an armed critter wants to shoot more than a limited number of bullets, oldest deleted first
  - When an armed critter is gone it can notify its bullets

Destruction

```
cCritterBullet::~cCritterBullet()
{
     if(_pshooter)
          _pshooter->removeBullet(this);
}

cCritterArmed::~cCritterArmed()
{
    for(int i = 0; i<_bulletarray.GetSize(); i++)
          _bulletarray.GetAt(i)->_pshooter = NULL;
}
```