3D Shooting Games

CS134

Chris Pollett

Oct 25, 2004.

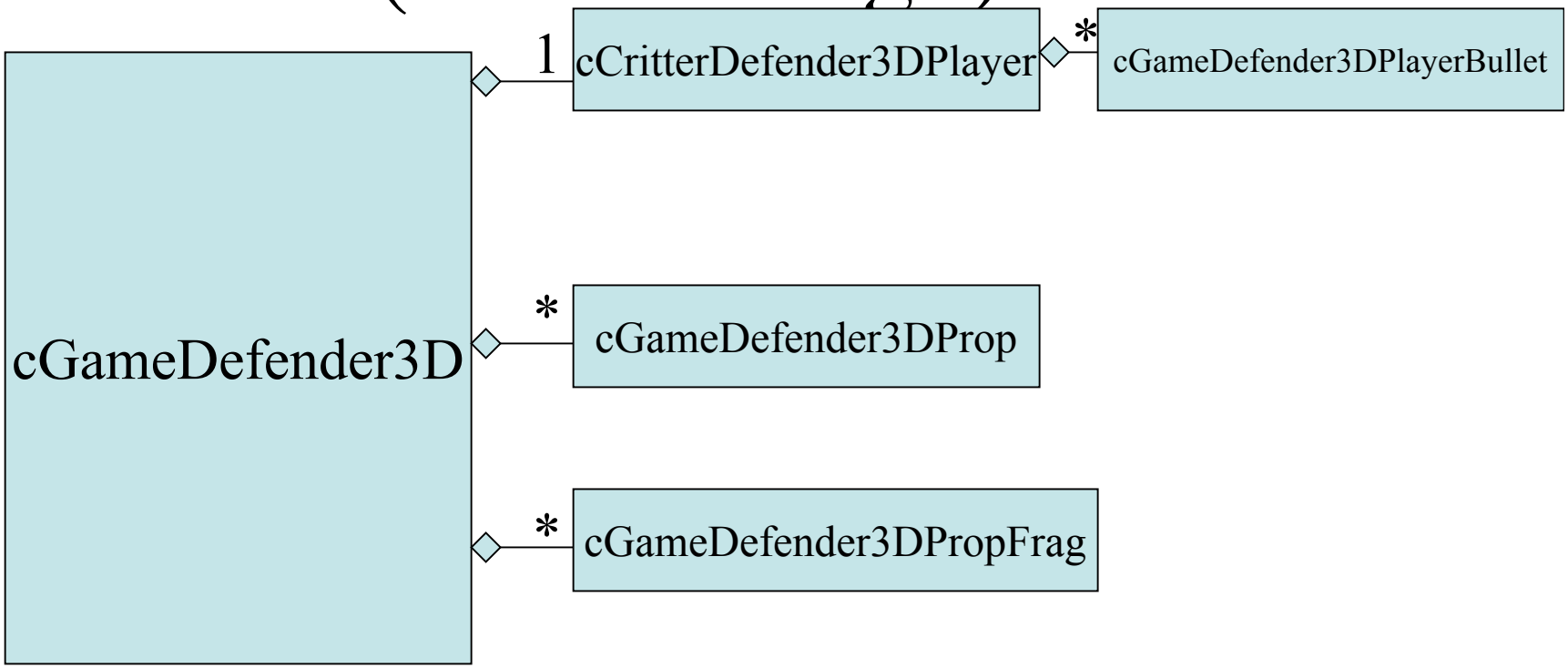
Outline

- Defender 3D
  - Specification
  - Design
  - Code

Defender 3D Spec

- Concept
  - Polygons rush at you, you have to shoot them before they pass you by and hit the back wall
- Appearance
  - First person 3D view, circle at center of screen where shooting at. When something hits it becomes a round polygon bouncing at bottom of screen. Sky bitmap covers the far wall.
- Controls
  - Left, right, up, down arrows. Move in and out with PgUp/PgDown. Spacebar shoots
- Behavior
  - Each polygon hit adds 10 to score. Hitting a polygon releases a shower of coins. Bumping into a coin kills it and adds one to player health. Coins evaporate after 3 seconds

Defender3D UML Diagram (Class Design)



cGameDefender3D Code

- Overrides in cGameDefender3D fairly routine.
- The constructor does _border.set(19, 19, 41). Third coordinate give z-thickness.
- seedcritters like in Spacewar
- adjustGameParameters replenishes the world with critters if it has fallen below _seedcount many.
- initializeView(CPopView *pview) call sets the viewer to use a cListenerViewerRide
- initializeViewpoint(cCritterViewer *pviewer) tweaks viewer depending on type of listener

cGameDefender3DPlayer Code

- Constructor gives player a new kind of listener: a cListenerArrowAttitude.
- This maps left/right ,up/down PgUp/PgDn to move the critter along normal, binormal, and tangent directions.

Setting up the sight to shoot at

```
setAttitudeToMotionLock(FALSE);
setAimToAttitudeLock(FALSE);
setAttitudeTangent(-cVector::ZAXIS);
setSprite(new cSpriteCircle());
psprite()->setFilled(FALSE);
psprite()-
   >setSpriteAttitude(cMatrix::yRotation(PI/2.
   0));
```

Shooting

- It turns out to be hard to shoot in 3D. So to make it easier nearest critter to where shooting at labelled a target and a cForceObjectSeek added to bullet:

```
cCritterBullet* cCritterDefender3DPlayer::shoot()
{
    playSound("Gunshot");
    cCritterBullet *pbullet = cCritterArmed::shoot();
    cCritter *paimtarget = pgame() ->pbiota()-
       >pickClosestTargetAhead(cLine(position(), aimvector()), this);
    pbullet->addForce(new cForceObjectSeek(paimtarget, 20.0));
    return pbullet;
}
```

collide

```
BOOL cCritterDefender3DPlayer::collide(cCritter *pcritter)
{
    BOOL collideflag = cCritter::collide(pcritter);
    if(collideFlag && pcritter-
       >IsKindOf(RUNTIME_CLASS(cCritterDefender3DPropFrag))
    {
        playSound("Ding");
        setHealth(health()+1);
        pcritter->die();
    }
}
```

collidesWith

```
BOOL cCritterDefender3DPropFrag::collidesWith(cCritter
    *pcritterother)
{
     if(pcritterother == pplayer())
         return cCollider::COLLIDESARG;
     else
         return cCollider::DONTCOLLIDE;
}
```

cGameDefender3DProp Code

- Makes props spin with:

randomizeSpin(1.0, 5.0)

- Also, adds a force of gravity in z axis
- Want props to start out at far end of world and move toward player:

```
randomizePosition(cRealBox(_movebox.locorner
    (), _movebox.hicorner() - (1.0
    -.2)*movebox.zsize()*cVector::ZAXIS);
```

Prop update

```
void cCritterDefender3DProp::update(CPopView
   *pactiveview)
{
    cCritter::update(pactiveview);
    if(_outcode & BOX_HIZ)
    {
        playSound("BONK");
        pplayer->damage(1);
        delete_me();
        return;
    }
}
```

Prop Die

```
void cCritterDefender3DProp::die()
{
    playSound("Explosion");
    for(int i=0; i<cCritterDefender3DProp::FRAGCOUNT;
       i++)
          newcCritterDefender3DPropFrag(this);
    _age = 0.0;
    setUsedFixedLifetime(TRUE);
    setShield(TRUE);
    setAttitudeToMotionLock(FALSE);
    setSpin(0.0);
    …
```