2D Shooting Games

CS134

Chris Pollett

Oct 20, 2004.

Outline

- The Spacewar game
  - Specification
  - Design
  - Spacewar code
- The 2D Game Stub
- The Worms games

The Spacewar game

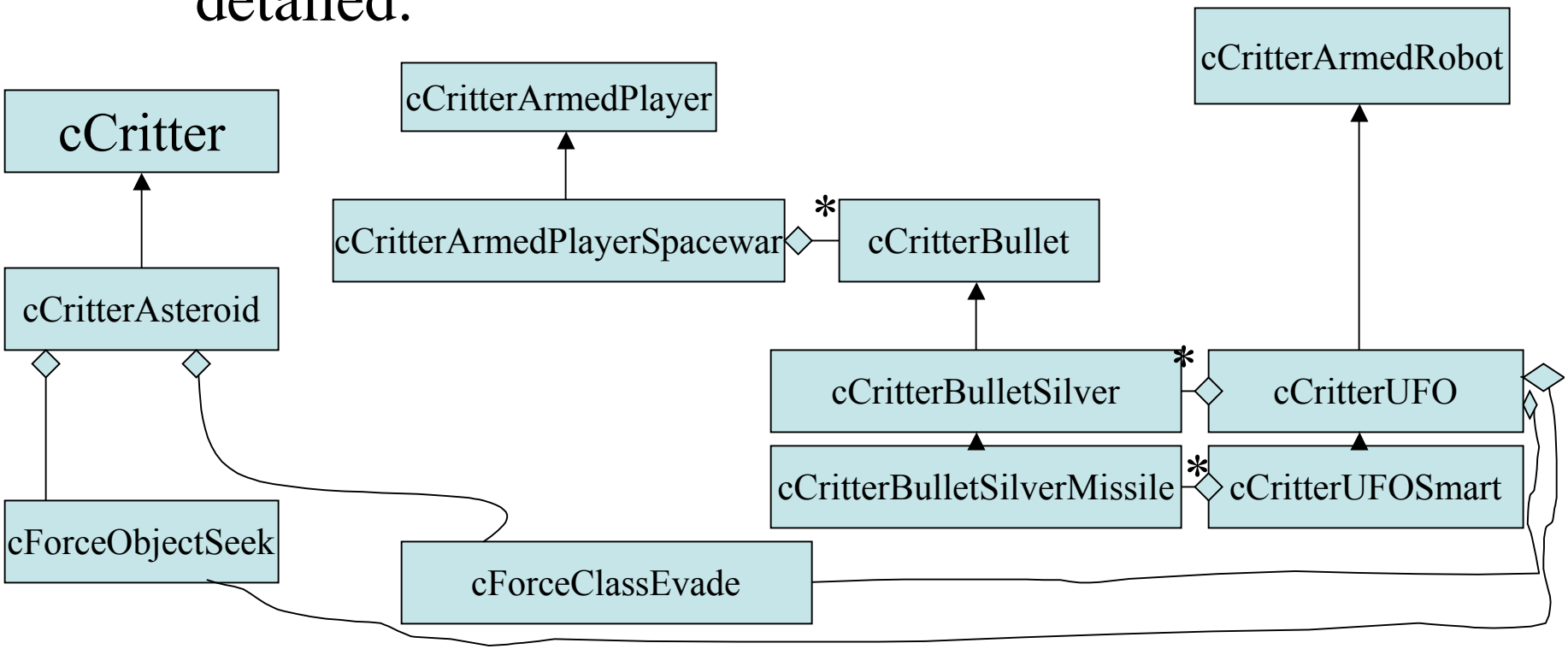
- Spacewar was the first computer game
  - Real version was 2-player, written in 1962 for a PDP1
- Pop framework game is more like Asteroids.

Specification

- Concept
  - Player tries to shoot and avoid asteroids. Occasionally, UFOs shoot at player. Asteroids try to avoid bullets.
- Appearance
  - This should be some doodled drawings of what game should look like
- Controls
  - Game uses Spaceship listener. Up and down accelerate ship, left and right rotate it.
- Game Play
  - When all asteroids killed a fresh wave appears which is faster.
  - You lose health when you hit an asteroid or get hit by a UFO
  - Your own bullets can't hurt you
  - Get points when an asteroid disappears from the screen.
  - The point break down for killing things is: asteroid 4, UFO 6, green enemy bullet 4, blue missile 8
  - Player health goes up by one point for every 100 point increase in score
  - Every 40 points a UFO appears

Design

- UML diagram
  - Should be simple reasonably but not too detailed:



Draft of Header

- Useful to get some idea on what things to override,etc:

```
class cCritterArmedPlayerSpacewar : public cCritterArmedPlayer
{
    public:
      cCritterArmedPlayerSpacewar(cGame *pownergame = NULL);
    void reset();
};
class cCritterAsteroid : public cCritter
{
    public:
      cCritterAsteroid(cGame *pownergame = NULL);
      virtual int damage(int hitstrength);
}; //etc
```

Code

- Once have gotten this far then could try to code things.
- Here are some highlights about the spacewar game
  - cGameSpacewar constructor makes the _border square and gives it a black background
  - cCritterArmedSpacewar critter is implemented so can adjust player's _health, _newlevelscorestep, _newlevelreward, etc. It sets the color for player sprite and sets _lastinvasionscore to 0.
  - seedCritters gets rid of any asteroid or bullets, but leaves UFOs alone. It adds back in _seedCount asteroids with the code:

```
for(int i = 0 ; i < _seedcount; i++)
{
  new cCritterAsteroid(this);
}
```

  - Idea on leaving UFOs alone is that if clearing level causes a UFO want to let that one alive

More on code

- cGameSpacewar's adjustGameParams:
  - Ends the game if the player's health is gone
  - Reseeds the screen with asteroids if all the asteroids and UFOs are dead; also speeds game up
  - Adds a new UFO every fixed increase in score.
- To generically make critters move faster at the end of a level the cCritter::MAXSPEED value is increased

Yet more on code

- Except for some initialization cCritterArmedPlayerSpacewar is almost the same as cCritterArmedPlayer.
- cCritterAsteroid, cCritterUFO, and cCritterUFOSmart require more work
- They use respectively the sprites: cPolygon, cPolyPolygon and cSpriteIcon.

Asteroid constructor

```
cCritterAsteroid::cCritterAsteroid(cGame *pownergame): cCritter(pownergame)
{
    setHealth(cCritterAsteroid::HEALTH);
    setValue(cCritterAsteroid:: VALUE);
    if(pownergame)
        setSprite(pgame()->randomSprite(pownergame->spritetype()));
    randomize(cCritter::MF_VELOCITY | cSprite::MF_RADIUS);
    psprite()->setLineColor(cColorStyle::CN_WHITE);
    addForce(new
      cForceClassEvade(cCritterAsteroid::DARTACCELERATION,
      cCritterAsteroid::DARTSPEEDUP, RUNTIME(cCritterBullet), FALSE));
    moveToMoveBoxEdge();
    if(pownergame)
        addForce(new cForceObjectSeek(pplaye(),
          cCritterAsteroid::CHASEACCELERATION));
}
```

Even more code

- The only differences between UFOs and UFOSmart are: the latter have different sprites, the latter are twice as fast, and the missiles of the latter steer toward player and bounce off edge of screen.
- The code for splitting asteroids that are hit is in damage:

```
int cCritterAsteroid::damage(int hitstrength)
{
        if(_shieldfield || recentlyDamaged()) return 0;
        int deathreward = cCritter::damage(hitstrength);
        playSound("Ding");
        if(_health)
        {
            setRadius(radius()/sqrt(2.0));
            mutate(cCritter::MF_NUDGE);
            if(pownerpiota()->count(RUNTIME_CLASS(cCritterAsteroid))<
                    cCritterAsteroid::OVERPOPULATIONCOUNT)
                    replicate();
        }
        return deathreward;}
```

2D Game Stub

- Gamestubs are meant to be possible starting points for your games
- Idea is to have five kinds of critters: the player, the player's bullets, a rival armed critter, its bullets, and a prop critter that might act as food.
- cGameStub has a _rivalcount in addition to a _seedcount from cGame

Stub's seedcritters

```
void cGameStub::seedCritters()
{
    pbiota()->purgeNonPlayerWallCritter();
    for(int i=0; i < _seedcount; i++)
        new cCritterStubProp(this);
    for (i=0; i<_rivalcount; i++)
        new cCritterStubRival(this);
}
```

Making Prop's Healthy

```
BOOL cCritterStubPlayer::collide(cCritter *pcritter)
{
    BOOL collideflag = cCritter::collide(pcritter);
    if(collideflag && pcritter-
       >IsKindOf(RUNTIME_CLASS(cCritterStubProp)))
    {
        setHealth(health()+1);
        pcritter->die();
    }
    return collideflag;
}
```

The Worms Game

- Worms is a test game for a bunch of different things in Pop.
- Worms are made up of cCritterWormSegments that use cForceObjectSpringRod forces to stay together.
- The player's sprite illustrates animation loops with cSpriteLoop.
- Worm bullet's run away from the player and never notice any other critter
- Eating these bullets is healthy
- These bullets have a lower priority than player's
- Rivals get smaller when bump into worm segments and grow when hit by player bullets