Selection Games

CS116A

Chris Pollett

Nov. 8, 2004.

Outline

- PickNPop Demo
- Specification
- Design
- Implementation
- Other Selection Games

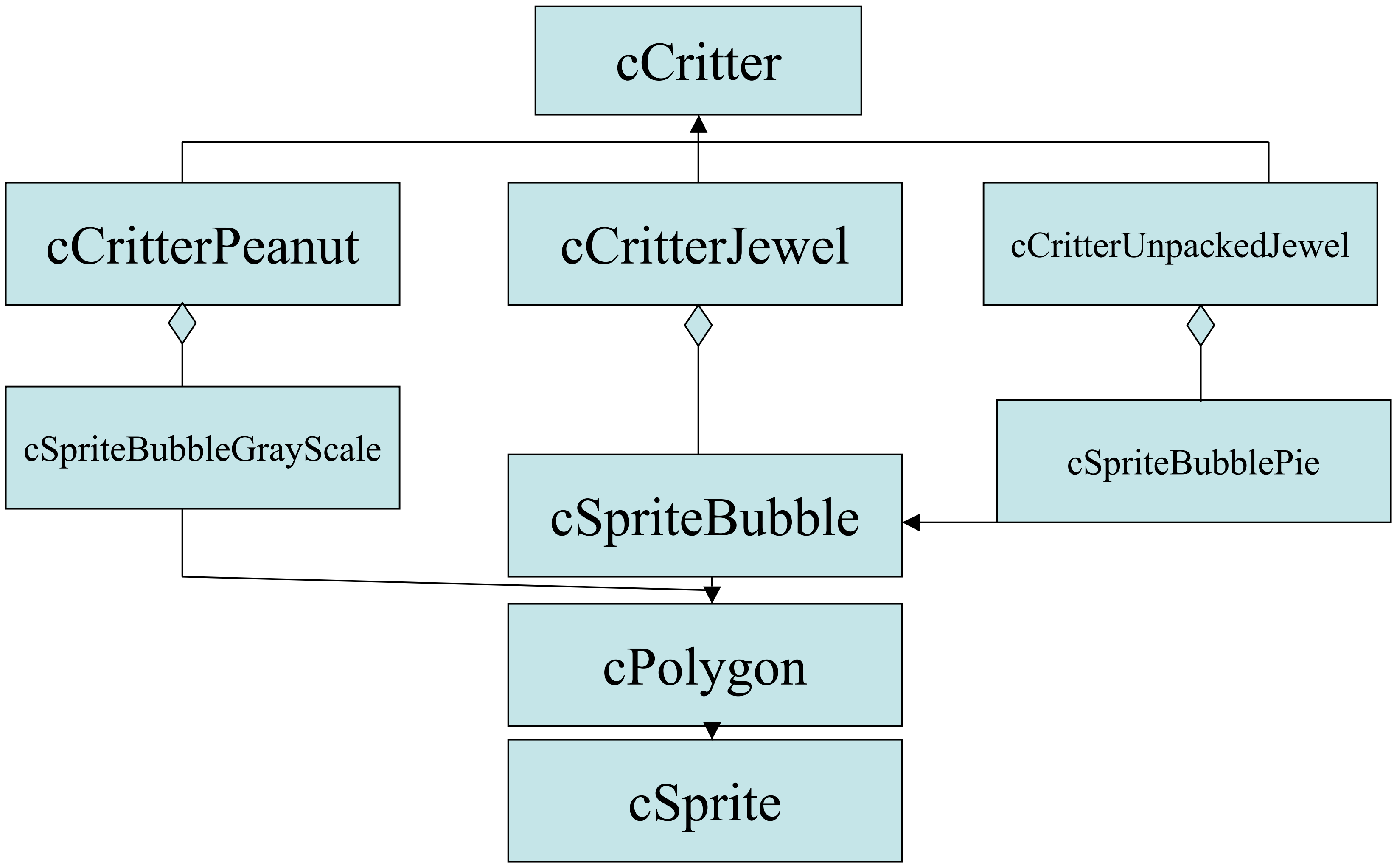
Specification

- Concept -- race against time to unpack jewels from a box. Have white and colored bubbles. Colored bubbles are the valuable jewels. Want to pop the white bubbles to get at colored ones.
- Appearance -- some pictures.
- Controls -- mouse uses one of two kinds of cursor tools: a popping tool and a dragging tool. Can use mouse wheel or toolbar to select tool.
- Behavior -- Screen is divided into two parts. At start all disks to the left. Need to drag jewels to right; pop blocking white bubbles. 1000pts for completing round. Get points for moving each jewel or popping non-jewel. Lose points for popping a jewel.

Design

- Player in this games is offscreen.The player critter is used as a container to hold the game score.
- Game has two cGraphicsRealBox2's: _packingbox and _targetbox. Bubble/jewels start in the _packingbox.
- New kinds of critters are used for jewels, non-jewels (called peanuts), and unpacked jewels.
- The constructors of each are overridden to give them the appropriate sprites.
- The die() methods of these classes have also been overridden to add a sound and to add _value to player's score
- Finally, cCritterJewel::update has been overridden to detect if critter is in _targetbox. If so, it replaces itself with cCritterGoodJewel.

UML


cCritter
cCritterPeanut
cCritterJewel
cCritterUnpackedJewel
cSpriteBubbleGrayScale
cSpriteBubble
cSpriteBubblePie
cPolygon
cSprite

Implementation

- Unlike Spacewar and Airhockey, _border of the world has a nonzero z size so that the shapes can pass above and below each other when game played in 3D.
- Implementation of scoring a little tricky -- want score for completing a round always 1000 but want to be able to vary the number of bubbles. JEWEL_PERCENT controls percent of world covered in bubbles.
- seedBubbles repsonsible for adding bubbles. Also figures out how much each bubble is worth and a value for _scorecorrection.
- seedCritters computes peanutstoadd and jewelstoadd. Add jewels first, so when painter algorithm applied they will be buried in MFC. In OpenGL cGame::zStackCritters used to achieve this effect.

seedCritters

```
void cGamePickNPop::seedCritters()
{
    int i;
    int jewelstoadd, peanutstoadd;
    Real jewelprobability = cGamePickNPop::JEWEL_WEIGHT;
    int jewelvalue(0), peanutvalue(0);
    cCritter *pcritternew;
    jewelstoadd = int(jewelprobability*_seedcount);
    peanutstoadd = _seedcount -jewelstoadd;
    jewelvalue =
        int(_maxscore*cGamePickNPop::JEWEL_GAME_SCORE_WEIGHT)/(jewelstoad
        d?jewelstoadd:1);
    peanutvalue = int(_maxscore -
        jewelstoadd*jewelsvalue)/(peanutstoadd?peanutstoadd:1);
    _scorecorrection = _maxscore -(jewelstoadd*jewelsvalue+peanutstoadd*peanutvalue);
    …
```

More seedCritters

```
    ….
    _pbiota->purgeNonPlayerNonWallCritters();
    for(i=0; i<peanutstoadd; i++)
    {
        pcritternew = new cCritterPeanut(this);
        pcritternew->setValue(peanutvalue);
    }
    for(i=0; i<jewelstoadd; i++)
    {
        pcritternew = new cCritterJewel(this);
        pcritternew->setValue(jewelvalue);
    }
    zStackCritters();
}
```

The World Rectangles

- We want the PickNPop game to fit as nicely as possible within window.
- So CDocument is given a cGraphicRealBox _packingbox and _targetbox which are supposed to fit within _border.
- The actual values are calculated in terms of _border.
- cRealBox::innerBox is used to get a box slightly within _border.
- Finally, colors for boxes set.

Converting a critter using update

```
void cCritterJewel::update(CPopview *pactiveview)
{
    cGamePickNPop *pgamepnp = NULL;
    cCritter::update(pactiveview);
    cVector safevelocity(_velocity);
    safevelocity.setZ(0.0);
    setVelocity(safevelocity);
    if(pgame()->IsKindOf(RUNTIME_CLASS(cGamePickNPop)))
        pgamepnp = (cGamePickNPop*)(pgame());
    else
        return;
    cRealBox effectivebox = pgamepnp-
       >targetbox().innerBox(cGamePickNPop::JEWELBOXTOLERAN
       CE*radius());
    ….}
```

More update

```
    …
    if(!effectivebox.inside(_position)) return;
    playSound("Ding");
    cCritterUnpackedJewel *pcritternew = new
        cCritterUnpackedJewel(this);
    pcritternew->setMoveBox(pgamepnp->targetbox());
    pcritternew->setDragBox(pgamepnp->targetbox());
    delete_me() // make a service request
    pcritternew->add_me(_pownerbiota); //another service request
    pgamepnp->pplayer()->addScore(_value);
}
```

Other Selection Games

- How would you implement Simon or some other memory game?
- Book suggests if doing a memory game with cards to override draw and then based on a flag draw a cover for a card or not.