Sprites

CS134

Chris Pollett

Oct 4, 2004.

Introduction

- Kinds of sprite
- The cSprite class
- Polygons
- Composite Sprites
- The cSpriteIcon class
- cSpriteLoop and cSpriteDirectional

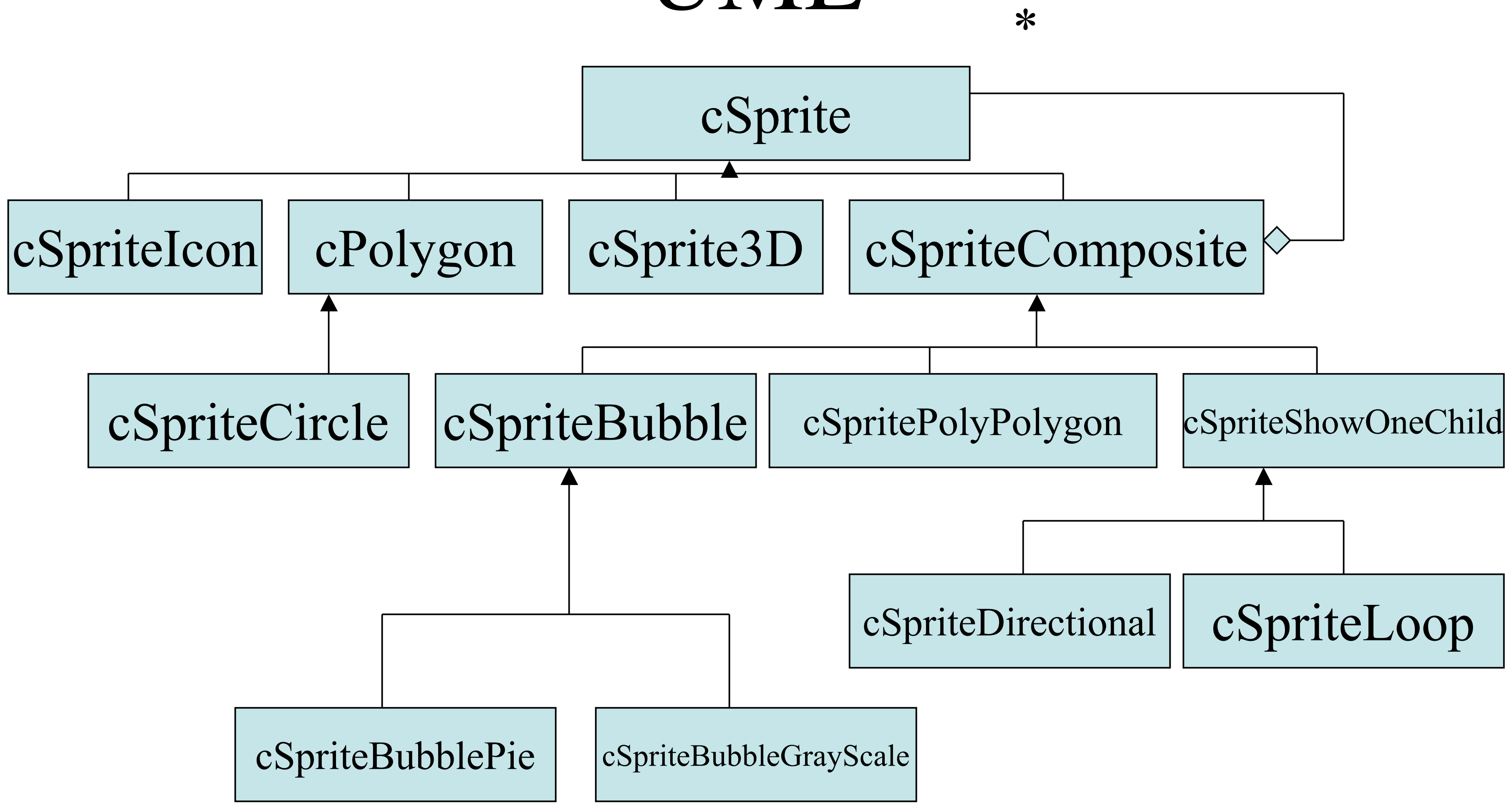
Kinds of sprite

- ‘Sprite’ in computer games just means a little character you can move around. Not some fairy creature from D&D.
- Often based on bitmaps (In Pop, SpriteIcon).
- Can be built based on geometrical objects to make scale independent. Ex. cPolygon.
- Can make complicated sprites from simple ones: cSpriteComposite which has children: cSpriteBubble, or cPolyPolygon.

More kinds of Sprites

- One variant of cSpriteComposite is cSpriteShowOneChild. Could use if wanted animations (cSpriteLoop) or the sprite to change appearance depending on the direction one is moving (cSpriteDirectional).

UML


*
cSprite
cSpriteIcon
cPolygon
cSprite3D
cSpriteComposite
cSpriteCircle
cSpriteBubble
cSpritePolyPolygon
cSpriteShowOneChild
cSpriteDirectional
cSpriteLoop
cSpriteBubblePie
cSpriteBubbleGrayScale

The cSprite class

- Does **not** have a pointer to its owner critter
  - So don't have to maintain any inverse reference
- Has a Real _radius to specify its size.

- Since sprites might be composite, the method:

  virtual Real cSprite::radius() shouldn't always return the value _radius.

  This method is used for collision detection
- Sprites also have a _spriteattitude matrix which by default is the identity matrix

The sprite draw Method

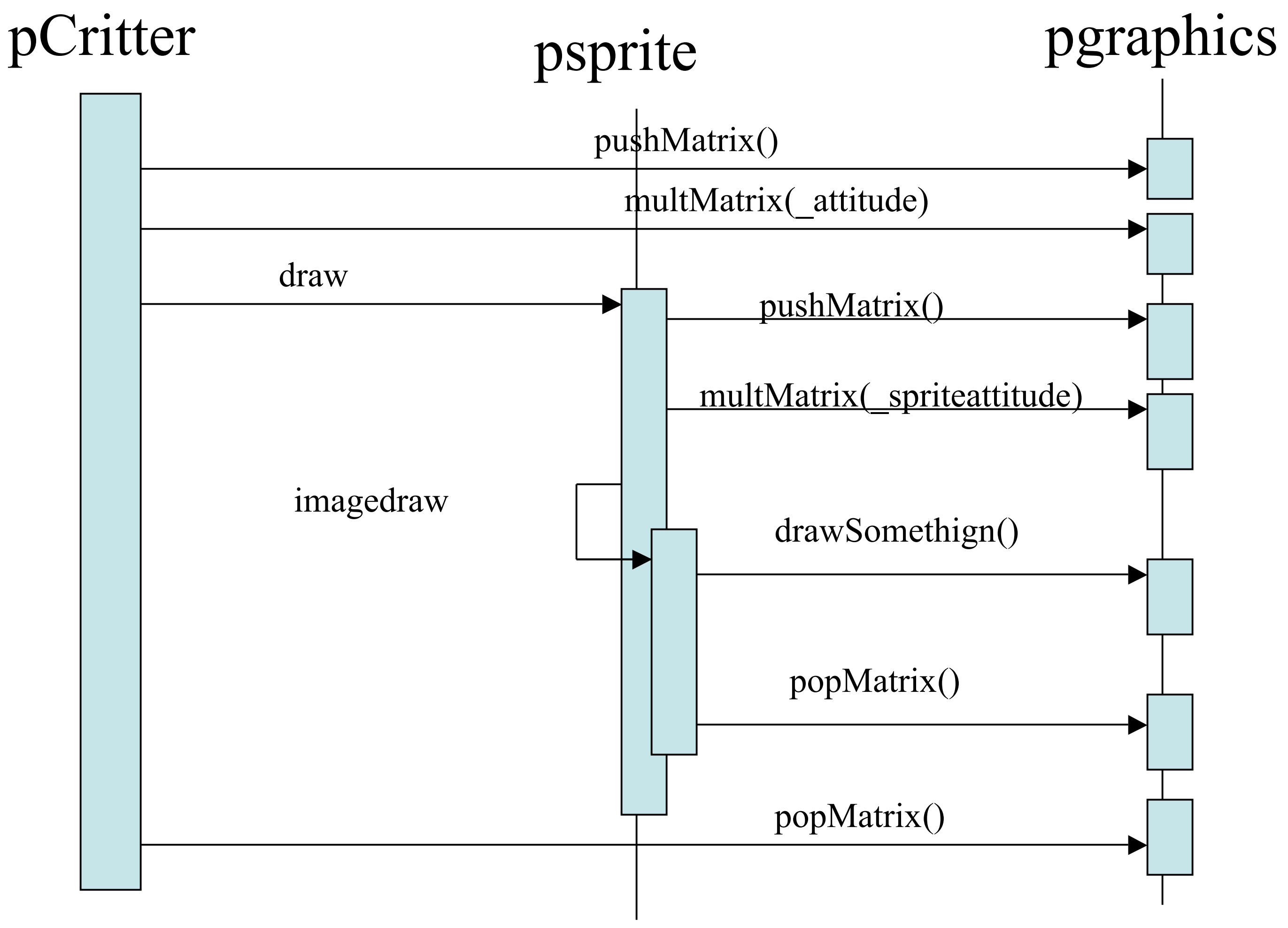
- Has the same arguments as cCritter::draw
- Manipulates graphics matrices and call helper method cSprite::imagedraw

- Graphics pipeline gets critter's _attitude
  - Moves zero vector to critter origin
  - Sets Sprite spatial attitude to match critters.
    - Should Multiply by _spriteattitude if want to position some other way

Graphics Pipeline

- The graphics pipeline is implemented as a cGraphics object which maintains two cMatrix's
  - One is the projection matrix
  - The other is the modelview matrix.
    - This has the form MV = V'*Mc*Ms where Ms is the _spriteattitude, Mc is the critter's _attitude, and V' is the _attitude of the cCritterViewer which views the scene.
    - In case of composite sprites might have subsidiary matrices Msa multiplies to the right of Ms
- A vertex u of a sprite polygon is transformed to: u'= P*MV*u where P is projection matrix.
- Multiplication is done with cGraphics::multMatrix

Sequence Diagram of draw



imagedraw

- The above diagram implements the template pattern where specific drawing is done by the imagedraw function. For example

  ```
  void cPolygon::imagedraw(cGraphics *pgraphics, int
     drawflags)
  {
     pgraphics->drawpolygon(this, drawflags);
  }
  ```

- For cSpriteIcon::imagedraw the call pgraphics->drawbitmap(this, drawflags); is done
- Remember cGraphics is a bridge to underlying MFC or OpenGL implementation

The animate method

- cCritter::animate(dt) does two things:
  - Makes an updateAttitude(dt) call to
    - match the critter’s attitude to the critter’s current motion matrix if the critter’s _attitudemotionlock is TRUE, or, otherise
    - rotate the critter’s _attitude by dt*_spin or
    - leave the the _attitude alone if _spin is zero
  - Calls _psprite->animate(dt,this)
    - Does nothing by default. Could look at powner->recentlyDamaged() and change sprite or look at dt and change sprite size, etc.

More animate

- cGraphicsMFC needs different bitmaps for different directions since can’t rotate bitmaps.
- cSpriteShowOneChild::_showindex says which sprite component is currently active
- cSpriteLoop::animate ages a timer and adjusts _showindex
- cSpriteDirectional::animate adjusts the _showindex sprite according to powner->tangent()

Polygons

- Most graphics systems have some way to draw polygons. Ex CDC::Polygon(POINT *vertices, int vertexcount) in Windows.
- Polygons scale well to different sizes.
- In Pop, can call cPolygon() to create an empty polygon.
- Then can use mutators to create a more intersting polygon

Polygon mutators

```
void setRegularPolygon(int vertexcount);
void setStarPolygon(int vertexcount, int step);
void setRandomStarPolygon(int mincount, int
   maxcount);
void setRandomRegularPolygon(int mincount, int
   maxcount)
void setRandomAsteroidPolygon(int mincount =5,
   int maxcount -30, Real spikiness = 0.3)

//The contructor cPolygon(n) creates a regular n-gon
```

More mutators

- polygon.h has more mutators for affecting polygon appearance
- ppolygon->randomize(cPolygon::MF_COLOR) can be used to randomize color MF_ALL to randomize all attributes
- Some attributes: _reallinewidth, _edged, _dotted, _realdotradius.
- cSpriteCircle is just a cSpritePolygon where the number of edges is large. Set by CIRCLESLICES.

Polygons in 3D

- When using cGraphicsOpenGL, polygons are drawn as a thick prism.
- The exact thickness of the prism is controlled by the Real _prismdz field

Composite Sprites

- cSpriteComposite holds an array of cSprite pointers called _childspriteptr.
- The default draw for this class looks like

```
for(int i=0; i<_childspriteptr.GetSize(); i++)
    _childspriteptr[i]->draw(pgraphics, drawflags);
```

cSpriteBubble

- cSpriteBubble ss implemented as a cSpriteComposite with two member sprite: a cSpriteCircle and a cSpritePolygon.
- The latter is supposed to be a fake reflection on the bubble.
- The reflections lives slightly on top of the circle to avoid “z-fighting”
- Doing this makes use of _spriteattitude and cMatrix::translation(cVector(side,0.5*side,.1)) to position this accent.

Polypolygons

- This is a cSpriteComposite which consists of a base polygon together with a secondary tipshape polygon for each vertex.
- setBasePoly(cPolygon* pppoly) and setTipShape(cSprite *pshape) can be used to set these.
- Spacewar give an example of polypolygons: Look at Game | Polypolygon.
- Tipshapes are rotated for each vertex.

The cSpriteIcon class

- Constructor is cSpriteIcon(int resourceID, BOOL transparent=TRUE, BOOL presetaspect= FALSE);
- Notice by default background of bitmap is transparent.
- This kind of sprite can be set with a line like
  - setSprite(new SpriteIcon(IDB_EARTH));
- The third argument of the constructor is used if one wants to fit sprites to some new rectangular shape. Ex: see cSpriteIconBackground
- To use this kind of sprite:
  - Need to create new .bmp files
  - Size them roughly according to how big will be onscreen.
  - Make edge size a power of 2. 16,32, 64 … this way will resize
  - Remember upper left pixel color is used as background color
  - Save bitmaps in 8 pixel mode
  - Use Project | Add Resource .. | Import … to add the resource

cSpriteLoop and cSpriteDirectional

- Are both arrays of other sprites
- Have ‘add’ method to add sprites.
  - For cSpriteIcon’s can add with add(resourceID)

  Ex:

  ```
  cSpriteLoop *pwalkman = new cSpriteLoop();
  pmanwalk->add(IDB_MAN1);
  pmanwalk->add(IDB_MAN2);
  setSprite(pmanwalk);
  ```

- The delay between flipping images set with cSpriteLoop::setFlipwait(Real flipwait)
- cSpriteDirectional splits circle into as many regions as add sprites. Shows nth sprite if moving in nth direction from clockwise vertical