Systémy častíc

Veronika Vlnková
2. ročník, informatika

A. Úvod

Systémy častíc je metóda využívaná v počítačovej grafike. Napriek svojej či práve vďaka svojej jednoduchosti majú systémy častíc široké využitie. Ideálne sú na simuláciu prírodných živlov ako dážď, sneh či oheň. Najprv boli používané vo filme, dnes sa pre svoju jednoduchosť veľmi často využívajú v hrách na nasimulovanie ohňa či dymu.
Na prvý pohľad by sa mohlo zdať, že je ich prínos obmedzený len na jednoduché animácie. Nie je však tomu tak. Svedčí o tom napríklad aj ich využitie vo filme Shrek. Spolu s ďalšími metódami boli použité na vytvorenie priam dokonalých efektov – na oživenie vodnej hladiny, pri ohnivých guliach, ktoré sršala dračica...
Záujemcu o hlbšie preniknutie do problematiky systémov častíc by som odporučila na [1].

B. Implementácia

Najprv sme si definovali triedu Vector. Ako už jej názov napovedá, trieda slúži na reprezentáciu dvojrozmerného vektora. Preťažením operátorov sme preň definovali všetky relevantné operácie ako priradenie, sčítavanie, skalárny súčin (násobenie), porovnanie na rovnosť a operácie k nim inverzné. Definícia triedy vyzerá takto.

```
class Vector {
  private:
    float x, y;
  public:
    Vector( float x = 0.0, float y = 0.0 );

    float GetX( void );
    float GetY( void );
    void SetX( float x );
    void SetY( float y );

    Vector operator = ( Vector v );
    Vector operator += ( Vector v );
    Vector operator -= ( Vector v );
    Vector operator *= ( Vector v );
    Vector operator /= ( Vector v );

    friend Vector operator + ( Vector v1, Vector v2 );
    friend Vector operator - ( Vector v1, Vector v2 );
    friend Vector operator * ( Vector v1, Vector v2 );
    friend Vector operator / ( Vector v1, Vector v2 );
    friend int operator == ( Vector v1, Vector v2 );
    friend int operator != ( Vector v1, Vector v2 );
  }
```

Particle (častica) je bod na obrazovke, ktorý má svoju pozíciu (pos), smer (dir) a životnosť (life). Každým zavolaním UpdateLife sa zníži životnosť častice o 1, pri nule častica zanikne. UpdateLife by mala byť volaná v každom frame, teda životnosť predstavuje počet framov, v ktorých sa častica objaví. UpdatePos mení pozíciu častice a to tak, že k nej pripočíta smerový vektor.

```
class Particle {
  private:
    signed long life;
    Vector pos, dir;
  public:
    Particle( signed long life = 0, Vector pos = Vector(), Vector dir =
              Vector() );
    signed long GetLife( void );
    Vector GetPos( void );
    Vector GetDir( void );
    void SetLife( signed long life );
```

```
    void SetPos( Vector pos );
    void SetDir( Vector dir );

    void UpdateLife( void );
    void UpdatePos( void );
}
```

ParticleArray je pole častíc. Sú preň definované klasické metódy pre prácu s poľom ako vkladanie do poľa, vymazávanie a metódy častíc vykonávané na celom poli naraz.

```
class ParticleArray {
  private:
    Particle *p;
    unsigned long num;
  public:
    ParticleArray( void );
    ~ParticleArray( void );

    unsigned long GetNum( void );
    Particle operator () ( unsigned long i );
    void Insert( unsigned long i, Particle p_ );
    void Delete( unsigned long i );
    void UpdateLife( void );
    void UpdatePos( void );
}
```

C. Snehové vločky

Sneh je naprogramovaný ako potomok poľa častíc.

Snehové vločky generujeme na oblohe aproximovanej úsečkou AB (určenou vektormi A, B), kde je časticiam na začiatku priradený smer kolmý na úsečku.
Pohyb častíc riadi metóda UpdateFrame - obnovuje pozíciu, smer, životnosť, vytvára nové častice, vymazáva mŕtve... Počet častíc v každom frame programu neprekračuje maxcount, nové častice vytvárame v počte framecount. Jednotlivým časticiam sa mení smer tak, aby sledovali jedinečný pohyb snehovej vločky, teda v našom prípade sledujú Brownov pohyb, častejšie nazývaný Random Walk. Pohyb častice sa dá ovplyvniť parametrom chaos, ktorý určuje ako chaoticky sa majú častice pohybovať - pre chaos rovný nule sa pohyb častíc zjednoduší na rovnomerné padanie po trajektórii určenej priamkou. Celkovo algoritmus simuluje jednotlivé padajúce snehové vločky tak, aby následne vytvárali dojem padajúceho snehu.

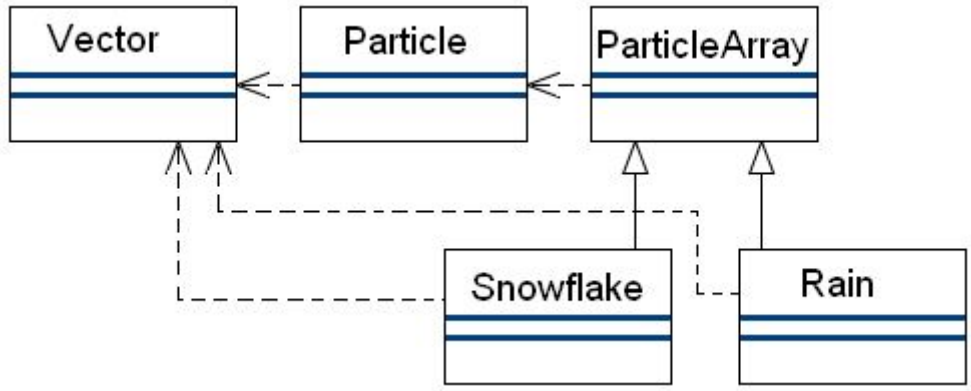
```
class Snowfall: public ParticleArray {
  private:
    Vector A, B;
    unsigned long framecount, maxcount, minlife, maxlife;
    float chaos, minspeed, maxspeed;
  public:
     void Initialize( Vector A, Vector B, float chaos, unsigned long
                      framecount, unsigned long maxcount, unsigned
                      long minlife, unsigned long maxlife, float
                      minspeed, float maxspeed );
    void UpdateFrame( void );
}
```

D. Dážď

Dážď je naprogramovaný podobne ako sneh. Rozdiely prirodzene vyplývajú z odlišností týchto dvoch živlov. Ťažké dažďové kvapky sú priťahované k zemi a nie nadnášané vzduchom, preto tu už nie je využitý Random Walk.

E. Diagram

Závislosti medzi triedami vyzerajú takto.



F. Použité programy

DevC++ - www.bloodshed.net

OpenOffice - www.openoffice.org

Irfanview - www.irfanview.com

Gimp - www.gimp.org

Zoznam použitej literatúry

[1] E. Ružický, A. Ferko: Počítačová grafika a spracovanie obrazu, Sapientia, Bratislava 1995