Информатика. Семинар №6

29.03.2016

https://dl.dropboxusercontent.com/u/96739039/sem4/infa_s06.pdf

Переход от бесконечного коэфф. трения к конечному

- Дополнительная корректировка для $\Delta\vec{p}_{\vec{\tau}}$:

$$m\Delta v_{\vec{n}} = \Delta\vec{p}_{\vec{n}} = \int_0^T \vec{N}dt$$

$$\Delta\vec{p}_{\vec{\tau}} = \int_0^T \vec{f}_{трения}dt, \left|\vec{f}_{трения}\right| \le \mu\left|\vec{N}\right|$$

$$\rightarrow \left|\Delta\vec{p}_{\vec{\tau}}\right| \le \mu\left|\Delta\vec{p}_{\vec{n}}\right|$$

P.S. знак у $\Delta\vec{p}_{\vec{\tau}}$ не должен измениться после этой корректировки

Парадигмы ООП

- Инкапсуляция (про public, private, protected)
- Наследование (дочерний класс перенимает возможности родительского)
- Полиморфизм (?)

Полиморфизм

- http://www.cplusplus.com/doc/tutorial/polymorphism/

virtual – виртуальный … действительный, фактический

Где это используется?

- Графический векторный редактор: много разных геометрических фигур, которые отрисовываются на экран.

```
struct Figure
{
  virtual void Draw() = 0;
};

struct Triangle: public Figure
{
  void Draw() { /* TODO */ }
};

struct Circle: public Figure
{
  void Draw() { /* TODO */ }
};
```

```
std::vector<Figure*> figures;

for (size_t i = 0; i < figures.size(); ++i)
{
  figures[i]->Draw();
}
```

Либо вы пишите стратегию…

```
struct Warrior
{
  virtual void Attack() = 0;
};

struct Archer : public Warrior
{
  void Attack() { /* TODO */ }
};

struct Knight : public Warrior
{
  void Attack() { /* TODO */ }
};
```

А если без полиморфизма…

```
std::vector<Triangle> triangles;
std::vector<Circle> circles;

for (size_t i = 0; i < triangles.size(); ++i)
{
  triangles[i].Draw();
}

for (size_t i = 0; i < circles.size(); ++i)
{
  circles[i].Draw();
}
```

Дублирование кода при добавлении новых фигур –> велика вероятность опечаток

Как работает полиморфизм

- Для каждого класса (не экземпляра, именно класса), имеющего хотя бы один ***virtual*** метод создаётся таблица виртуальных функций (vtable)
- Какую именно ф-ю надо запустить определяется не на этапе *линковки,* а в процессе работы программы (в run-time`e) – позднее связывание.
- Размер класса увеличивается на sizeof(void*) – указатель на соответствующую vtable
- Работа с виртуальными методами медленнее (около 10%), т.к. при каждом вызове требуется каждый раз искать указатель на нужный метод в vtable … но зачастую оно того стоит

https://habrahabr.ru/post/51229/

Виртуальный деструктор

```
struct A
{
  A() { a = new int[10]; }
  ~A() { delete[] a; }
  int* a;
};

struct B: public A
{
  B(): A()
  {
    b = new float[100];
  }
  ~B() { delete[] b; }
  float* b;
};

int main()
{
  A* b = new B;
}
```

- Деструктор класса B не будет вызван -> утечка памяти

Виртуальный деструктор

```
struct A
{
  A() { a = new int[10]; }
  virtual ~A() { std::cout << "~A"; delete[] a; }
  int* a;
};

struct B: public A
{
  B(): A()
  {
    b = new float[100];
  }
  ~B() { delete[] b; }
  float* b;
};

int main()
{
  A* b = new B;
}
```

virtual ~A(); ->
оба деструктора
будут вызваны

Наличие связей между телами

```
class IConstraint
{
public:
  Ball* balls[2];
  virtual void Handle() = 0;
};
```

```
class RigidConstraint : public IConstraint
{
  float length;

  void Handle()
  {
    // TODO
  }
};
```

```
class ViscoElasticConstraint : public IConstraint
{
  float length, k, alpha;
  void Handle()
  {
    // TODO
  }
};
```

P.S. Абстрактные классы иногда называют интерфейсами. Отсюда префикс “I” в названии класса.

Структура типовой игры

https://dl.dropboxusercontent.com/u/96739039/fb.zip

Домашнее чтение…

- http://www.parashift.com/c++-faq/