АЛГОРИТМЫ ПРОСТЫХ ИГР И СЕКРЕТЫ ИГРАБЕЛЬНОСТИ

Автор: Гурьянов Федор

Научный руководитель: Дединский Илья Рудольфович

ЦЕЛЬ И ЗАДАЧИ

Цель проекта:

- Рассмотрение алгоритмов простых игр, не требующих изощренного программирования, но при этом достаточно увлекательных.

Задачи:

- Как сделать игру интересной
- Рассматриваются реализация таких игр и идеи их создания
- Быстродействие
- Улучшение кода
- Рассматриваются готовые игры

КАК СДЕЛАТЬ ИГРУ ИНТЕРЕСНОЙ?

При 2D графике и неизощренном программировании игры на результат более интересные

Типы игр

- На прохождение (с конечной целью)
- Игры для режима мультиплеер (два или более игроков)
- На результат (с таблицей рекордов)

ЗОЛОТАЯ СЕРЕДИНА ПРИ СОЗДАНИИ ИГРЫ

При создании игры важно сделать ее хорошей и интересной, но при этом не пытаться сделать больше чем нужно или больше чем возможно

РЕАЛИЗАЦИЯ

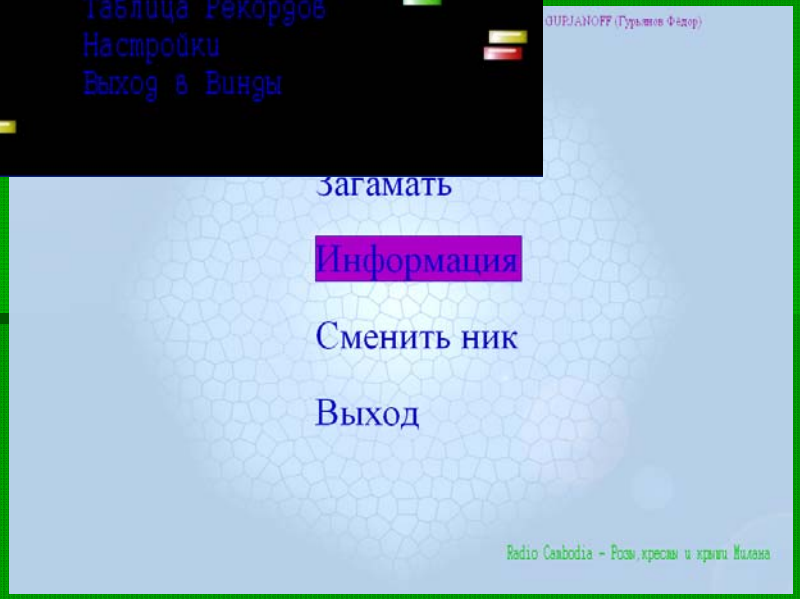
Идея

Алгоритм (самая простая реализация игры)

Улучшение внешнего вида

Усовершенствование (бонусы и т.д.)

Интерфейс



БЫСТРОДЕЙСТВИЕ

```
void Fonforgame (int lif, int score, int dlina, int dlina1, int dlina2, int* s)
{
HDC i = txLoadImage ("i.bmp");
HDC ii = txLoadImage ("ii.bmp");
HDC thr = txLoadImage ("tg.bmp");
HDC f = txLoadImage ("cht.bmp");
txBitBlt (txDC(), dlina2, 550, 1024, 768, thr, 0, 0);
txBitBlt (txDC(), dlina, 10, 1024, 768, f, 0, 0);
if (*s == 0) txBitBlt (txDC(), dlina1, 100, 1024, 768, i, 0, 0);
if (*s == 1)
  {
  txBitBlt (txDC(), dlina1, 100, 1024, 768, ii, 0, 0);
  *s = -1;
  }
txDeleteDC (i);
txDeleteDC (ii);
txDeleteDC (thr);
txDeleteDC (f);
}
```

Кривой код

```
void stack_manager::OnTimer ()
{
DrawStacks ();
hero.OnTimer ();
enemy.OnTimer (CD_HDC);
DrawFire ();
}
```

Правильный код

Просчет сложных моделей заранее (кэширование)

УЛУЧШЕНИЕ КОДА

- Массивы
- Структуры
- Классы

Без структур: void Fonforgame (int lif, int score, int col, int dlina, int dlina1, int size);

С структурами: void Fonforgame (hero* Hero, size* SizeObject);

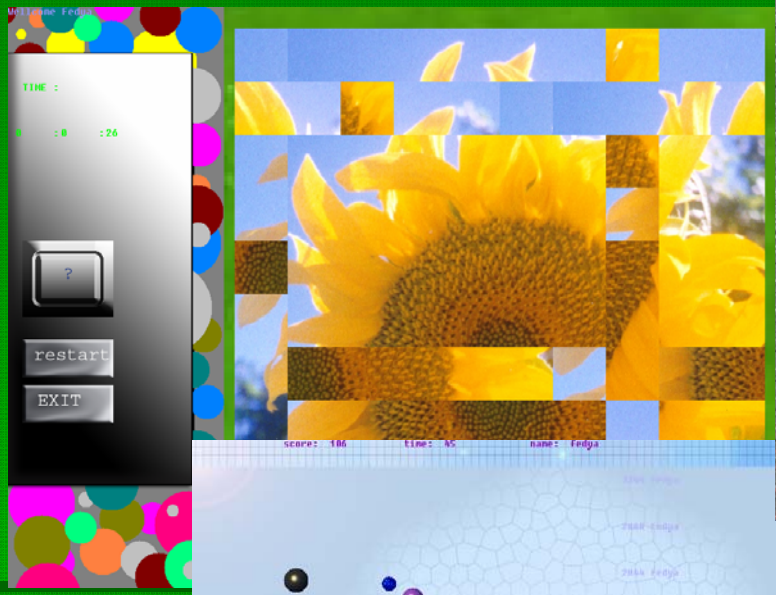
```
char name [20]      = "Fedya";
FILE* file = fopen ("name.fox", "r");
fscanf (file, "%s %d", &name, &Sound);
fclose (file);
PlaySnd ();
Shake               =  0;
int win             =  0;
HeroSize            =  40;
int _sleep          =  10;
int sizePOINTS      =  42;
Ball ball           = {200, 730, 0, 0, 15, -4, -4};
Point Hero          = {200, 745, 0, 2};
Point Points [42]   = {0};
Point Bonus_20      = {0, 200, 40};
Point BonusRand     = {200, 0, 40};
Bonuses BonusPlus   = {0,1000,0,0,0,1000,0,0,0,1000};
HDC Picture [3][3]  = {{txLoadImage ("11.bmp"), txLoadImage ("12.bmp"), txLoadImage ("13.bmp")},
                       {txLoadImage ("21.bmp"), txLoadImage ("22.bmp"), txLoadImage ("23.bmp")},
                       {txLoadImage ("31.bmp"), txLoadImage ("32.bmp"), txLoadImage ("33.bmp")}};
HDC fon             =  txLoadImage ("fon.bmp");
HDC bonus [3]       = {txLoadImage ("Bonus11.bmp"), txLoadImage ("Bonus12.bmp"),  txLoadImage ("Bonus13.bmp")};
HDC bonusPlus [3]   = {txLoadImage ("Live.bmp"),    txLoadImage ("HeroMove.bmp"), txLoadImage ("Score.bmp")};
HDC bonus2[3]       = {txLoadImage ("Bonus21.bmp"), txLoadImage ("Bonus22.bmp"), txLoadImage ("Bonus23.bm
LoadArray  (Points, sizePOINTS);
int Scorelimit = ScoreLimit (Points, sizePOINTS);
if (Scorelimit < 1092) Points [0].score += 1092 - 1082;
```

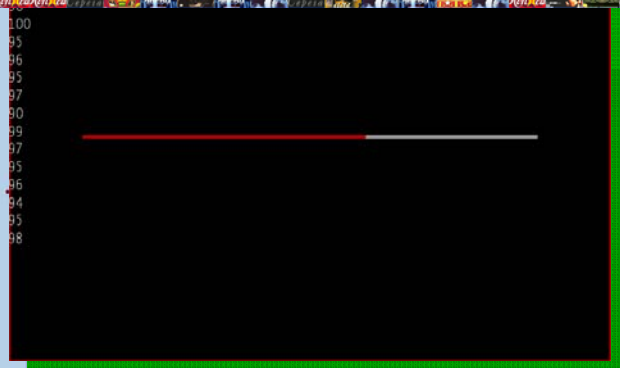
С классами

Без классов

```
void Draw ();
void Move ();

bool ok ();
void dump ();

Hero ();
Hero (const Hero& hero);
~Hero ();
};


class Enemy
{
public:

int draw_type_;
int x_coord_;
int speed_move_;

HDC Enemy_HDC [2];

void Draw ();
void Move ();
int OnTimer (HDC CD_HDC [10]);

bool ok ();
void dump ();

Enemy ();
Enemy (const Enemy& Enemy);
~Enemy ();
};


class stack_manager
{
public:
Hero hero;
Enemy enemy;

stack* stks_ [Size_stacks];
HDC CD_HDC [10];
HDC fon;
HDC panel;
```

ГОТОВЫЕ ИГРЫ





РЕЗУЛЬТАТЫ

- Мы рассмотрели простейшие алгоритмы игр.
- Обсудили их идеи и реализацию.