Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

## Лабораторная работа №2

«Процессы и асинхронное взаимодействие.»

Выполнили: студенты 4 курса

ИВТ, гр. ИП-712

Коротеев Максим   
Бахарев Максим

Проверил:

Заведующий кафедрой ПМиК

Фионов А.Н.

Новосибирск, 2020 г.

# Задание

1. Тщательно изучить библиотеку VinGraph.
2. Используя функции библиотеки VinGraph, нарисовать абстрактную картину, которой представлены (почти) все доступные графические элементы.
3. Заставить нарисованные элементы двигаться независимо друг от друга с помощью параллельных процессов (можно изменять во времени положение, цвет, размеры, конфигурацию графических элементов). Предусмотреть завершение программы по нажатию на любую клавишу.
4. Нарисовать нечто, движущееся по замкнутой кривой.

Организовать изменение траектории движения по нажатию на клавиши (организуя взаимодействие процессов через общую область памяти (shared memory)). В качестве фона можно

использовать (оживленную) картину, созданную на предыдущих этапах работы.

1. Затем последнюю программу сделать с помощью нитей в одном процессе.

# Выполнение заданий(1 и 2)

Была изучена библиотека VinGraph по соответствующей методичке.

Далее была разработана программа с использованием библиотеки vingraph.h по отрисовке всех доступных графических элементов.

Из-за ошибки в реализации дуги (Arc) после ее отрисовки происходит сбой в дальнейшей отрисовке и дальнейшее мгновенное завершение

программы. Поэтому сначала были отрисованы все элементы без Arc и затем отдельно сама дуга с 5 секундной задержкой после ее отрисовки.

На работу программы выделено 10 секунд, после которых происходит вызов метода CloseGraph() и программа закрывается.

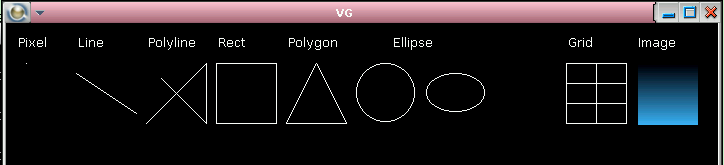


Рис.1 – графическое окно со всеми фигурами

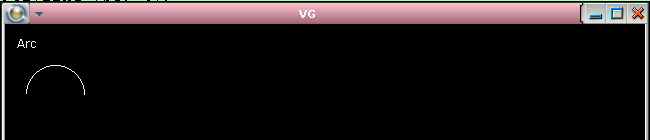


Рис.2 – графическое окно с отдельной отрисовкой дуги (Arc)

# Листинг программы

#include <stdlib.h> #include <unistd.h> #include <vingraph.h>

int main()

{

ConnectGraph(); Text(10, 10, "Pixel");

Pixel(20, 40);

Text (70, 10, "Line");

Line (70, 50, 130, 90);

Text (140, 10, "Polyline");

tPoint p1 [] = { {140, 100}, {200, 40}, {200, 100}, {155, 55} };

Polyline (p1, 4);

Text (210, 10, "Rect");

Rect (210, 40, 60, 60);

Text (280, 10, "Polygon");

tPoint p2 [] = { {280, 100}, {310, 40}, {340, 100} };

Polygon (p2, 3);

Text (385, 10, "Ellipse");

Ellipse (350, 40, 60, 60);

Ellipse (420, 50, 60, 40);

//Text (10, 10, "Arc");

//Arc (20, 40, 60, 60, 0, 1800);

//sleep(10);

Text (560, 10, "Grid");

Grid (560, 40, 60, 60, 3, 2);

int \*im\_buf = (int\*) malloc (60\*60\*4);

for (int i = 0, c = 10; i < 60; i++, c += 0x010304) for (int j = 0; j < 60; j++)

im\_buf [60\*i + j] = c; Text (630, 10, "Image");

Image32 (630, 40, 60, 60, im\_buf);

sleep(10); CloseGraph(); return 0;

}

**Выполнение заданий(3,4,5)**

Для выполнения 3 задания за основу был взял 1 пункт

лабораторной. Для выполнения движения была использована функция MoveTo(x, y, figure) – принимающая в себя координаты х и у и фигуру,

которую необходимо двигать. Далее для реализации параллельных процессов был использован метод fork() из библиотеки unistd.h. Было реализовано

вложенное создание процессов-родителей для реализации 4 разных процессов чтобы определить в каждый по 2 фигуры.

Так же с помощью InputChar() – было реализовано закрытие программы по нажатию любой из клавиш.

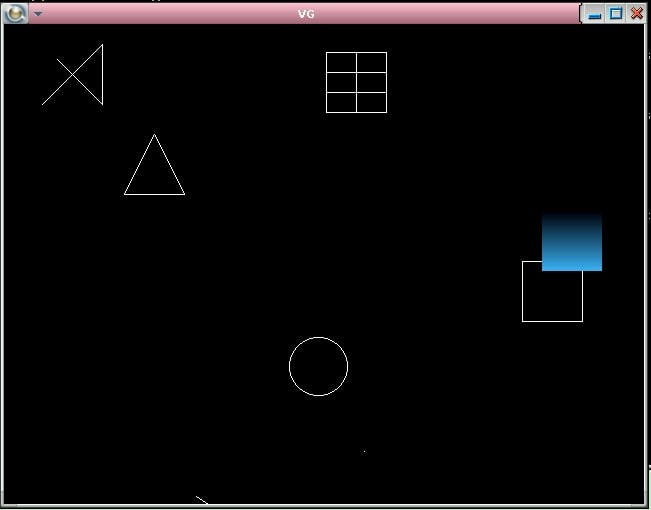
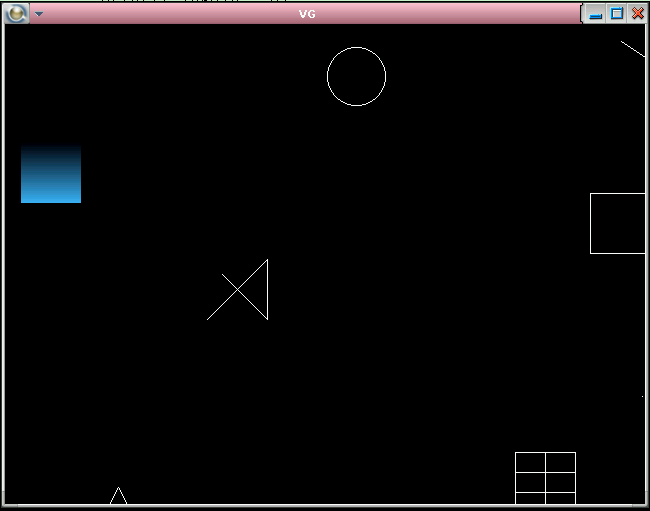


Рис.1 – демонстрация разных этапов работы программы

Для выполнения 4 задания была реализована программа с

использованием shared-memory, для инициализации такой памяти был

использован метод mmap с определенными флагами – MAP\_SHARED, MAP\_ANONYMOUS, влияющие на тип собственно памяти – разделяемая и анонимная.

Далее в программе было реализовано движение фигуры квадрат по Улитке Паскаля, теория которой была описана в задании к лабораторной. С помощью стрелок производится смещение «Улитки» в соответствующую

сторону, что продемонстрированно на рисунке 2.

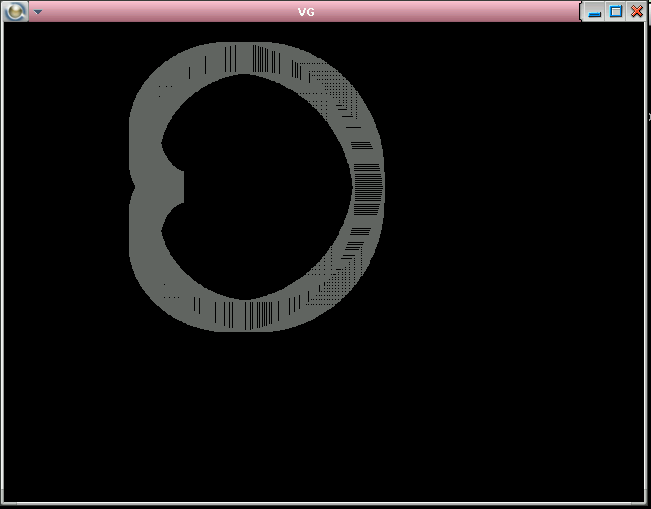


Рис.1

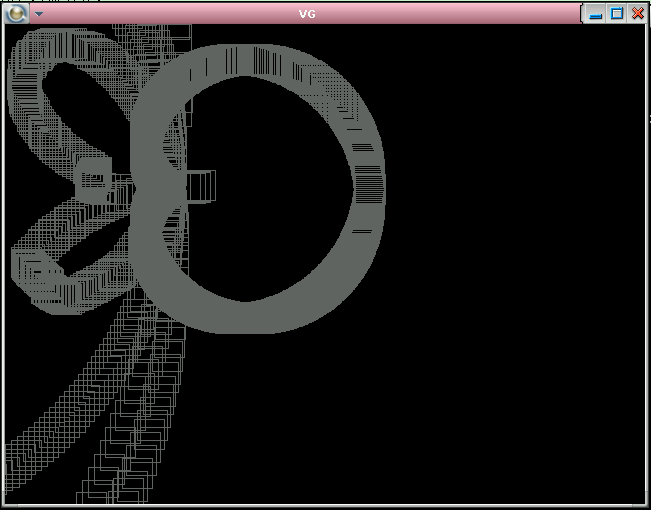


Рис.2

В 5 задании в программу были успешно внедрены нити из библиотеки <pthread.h>.

Циклы отрисовки и движения были разнесены в разные функции для нитей.

Далее в main() были инициализированы две нити – mover и drawer

соответственно. Потом была запущена нить mover, которую прерывает нить drawer при ее использовании.

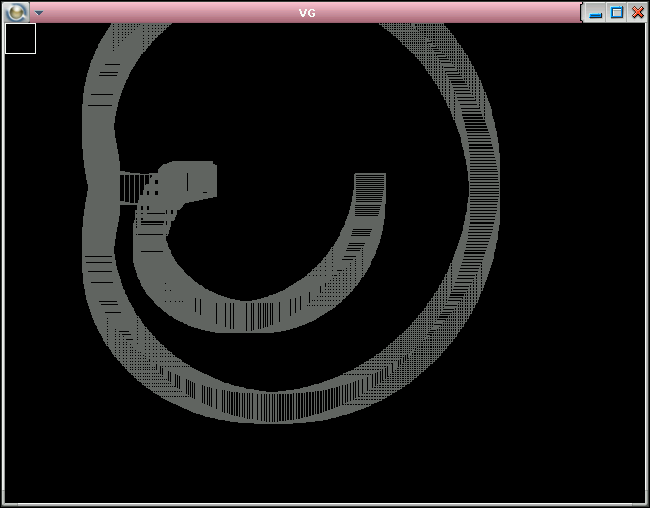


Рис.1 – демонстрация работы программы со сдвинутой улиткой паскаля.

**ПРИЛОЖЕНИЕ. ЛИСТИНГ ПРОГРАММ:**

1. cpp:

#include <iostream> #include <stdio.h> #include <stdlib.h> #include <sys/types.h> #include <unistd.h> #include <vingraph.h>

#include <time.h>

int main()

{

srand(time(0));

ConnectGraph();

int pixel = Pixel(20, 40);

int line = Line (70, 50, 130, 90);

tPoint p1 [] = { {140, 100}, {200, 40}, {200, 100}, {155, 55} };

int polyline = Polyline (p1, 4); int rect = Rect (210, 40, 60, 60);

tPoint p2 [] = { {280, 100}, {310, 40}, {340, 100} };

int polygon = Polygon (p2, 3);

int ellipse = Ellipse (350, 40, 60, 60);

int grid = Grid (560, 40, 60, 60, 3, 2); int \*im\_buf = (int\*) malloc (60\*60\*4);

for (int i = 0, c = 10; i < 60; i++, c += 0x010304) for (int j = 0; j < 60; j++)

im\_buf [60\*i + j] = c;

int im32 = Image32 (630, 40, 60, 60, im\_buf);

tPoint dim;

dim = GetDim(0); delay(1000);

int a = getpid(); printf("\n%d\n", a);

pid\_t proc1, proc2, proc3, proc4; if(proc1 = fork())

{

int b = getpid(); printf("\n%d\n", b); if(proc2 = fork())

{

int c = getpid(); printf("\n%d\n", c); if(proc3 = fork())

{

int d = getpid(); printf("\n%d\n", d); if(proc4 = fork())

{

}

else

{

InputChar();

int e = getpid(); printf("\n%d\n", e); CloseGraph();

srand(time(0)); while(true)

{

int way\_x = rand()%dim.x, way\_y

= rand()%dim.y;

rand()%dim.y;

}

}

}

else

{

MoveTo(way\_x, way\_y, pixel); way\_x = rand()%dim.x, way\_y =

MoveTo(way\_x, way\_y, line); delay(400);

srand(time(0)); while(true)

{

rand()%dim.y;

rand()%dim.y;

}

}

else

{

int way\_x = rand()%dim.x, way\_y =

MoveTo(way\_x, way\_y, polyline); way\_x = rand()%dim.x, way\_y =

MoveTo(way\_x, way\_y, rect); delay(600);

}

srand(time(0)); while(true)

{

}

}

else

{

int way\_x = rand()%dim.x, way\_y = rand()%dim.y; MoveTo(way\_x, way\_y, polygon);

way\_x = rand()%dim.x, way\_y = rand()%dim.y; MoveTo(way\_x, way\_y, ellipse);

delay(800);

}

srand(time(0)); while(true)

{

int way\_x = rand()%dim.x, way\_y = rand()%dim.y; MoveTo(way\_x, way\_y, grid);

way\_x = rand()%dim.x, way\_y = rand()%dim.y; MoveTo(way\_x, way\_y, im32);

delay(1000);

}

}

}

1. cpp:

CloseGraph(); return 0;

#include <cmath> #include <stdio.h> #include <stdlib.h> #include <sys/types.h> #include <sys/mman.h> #include <unistd.h> #include <vingraph.h> #include <time.h>

using namespace std;

int main()

{

ConnectGraph(); int key;

float\* gvara = static\_cast<float\*>(mmap(NULL, sizeof\* gvara, PROT\_READ|PROT\_WRITE,

MAP\_SHARED|MAP\_ANONYMOUS,

-1,0));

float\* gvarb = static\_cast<float\*>(mmap(NULL, sizeof\* gvarb, PROT\_READ|PROT\_WRITE,

MAP\_SHARED|MAP\_ANONYMOUS,

-1,0));

\*gvara = 100;

\*gvarb = 100; int x, y;

float rho, a, phi = 0,b;

pid\_t proc; if(proc = fork())

{

while(true)

{

key = InputChar(); if(key == ' ')

break; if(key == 81)

\*gvarb = \*gvarb + 5; if(key == 82)

\*gvara = \*gvara + 5; if(key == 83)

\*gvarb = \*gvarb - 5; if(key == 84)

\*gvara = \*gvara - 5;

}

}

else

{

CloseGraph(); return 0;

while(true)

{

Rect (x, y, 30, 30);

rho = \*gvara \* cos(phi) + \*gvarb; x = rho \* cos(phi) + 150;

y = rho \* sin(phi) + 150; phi += 0.01;

SetColor(RGB(100, 100, 100));

Rect (x, y, 30, 30); delay(15);

}

}

return 0;

}

1. cpp:

#include <cmath> #include <iostream> #include <stdlib.h> #include <pthread.h> #include <sys/types.h> #include <sys/mman.h> #include <unistd.h> #include <vingraph.h> #include <time.h>

using namespace std; float\* sh\_a;

float\* sh\_b; int x, y;

float rho, phi = 0;

void \*fig\_mover(void \*args)

{

static int key; while(true)

{

key = InputChar(); if(key == ' ')

break; if(key == 81)

\*sh\_b = \*sh\_b + 5; if(key == 82)

\*sh\_a = \*sh\_a + 5; if(key == 83)

\*sh\_b = \*sh\_b - 5; if(key == 84)

\*sh\_a = \*sh\_a - 5;

}

}

void \*fig\_drawer(void \*args)

{

while(true)

{

Rect (x, y, 30, 30);

rho = \*sh\_a \* cos(phi) + \*sh\_b; x = rho \* cos(phi) + 150;

y = rho \* sin(phi) + 150; phi += 0.01;

SetColor(RGB(100, 100, 100));

Rect (x, y, 30, 30); delay(15);

}

}

int main()

{

ConnectGraph();

sh\_a = static\_cast<float\*>(mmap(NULL, sizeof\* sh\_a, PROT\_READ|PROT\_WRITE,

MAP\_SHARED|MAP\_ANONYMOUS,

-1,0));

sh\_b = static\_cast<float\*>(mmap(NULL, sizeof\* sh\_b, PROT\_READ|PROT\_WRITE,

MAP\_SHARED|MAP\_ANONYMOUS,

-1,0));

\*sh\_a = 100;

\*sh\_b = 100;

pthread\_t mover, drawer;

pthread\_create(&mover, NULL, &fig\_mover, NULL); pthread\_create(&drawer, NULL, &fig\_drawer, NULL); if(!pthread\_join(mover, 0))

cout << "Code is" << pthread\_cancel(drawer); CloseGraph();

return 0;

}