ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

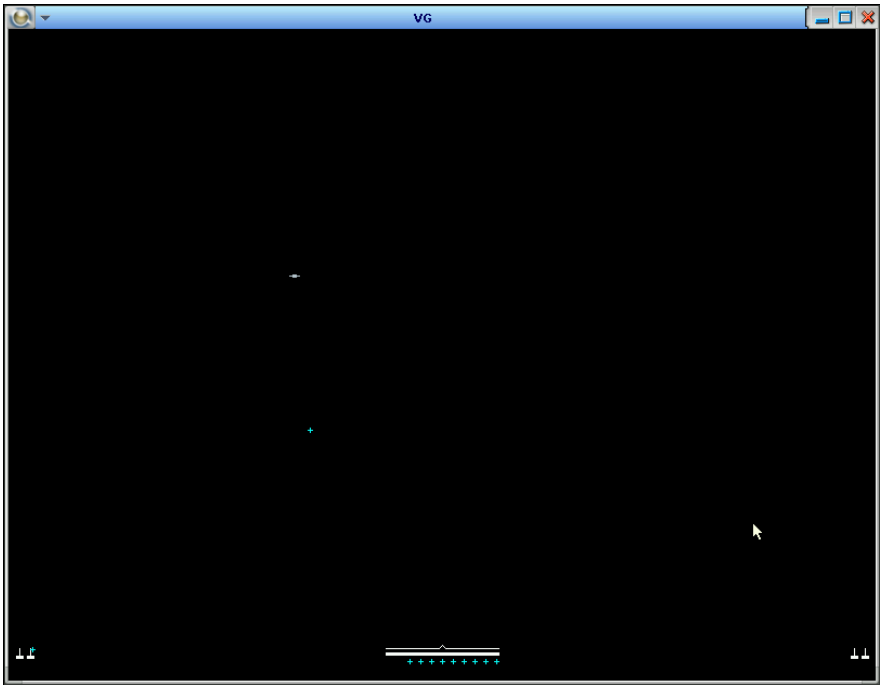
ЛАБОРАТОРНАЯ РАБОТА № 4

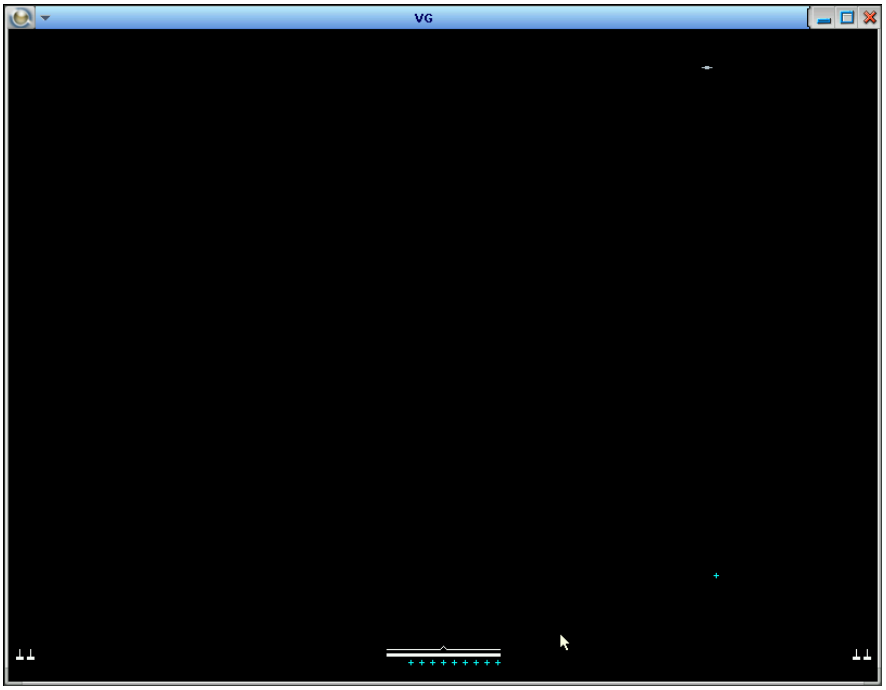
Выполнил:
Студент группы: ИП-712
Алексеев С.В.

Проверил: профессор кафедры ПМиК
Фионов А.Н.

Новосибирск – 2020

1. Написать программу, осуществляющую полет управляемого снаряда по квадрату 200x200, а затем по прямоугольнику 500x200 точек. Тарелок нет.



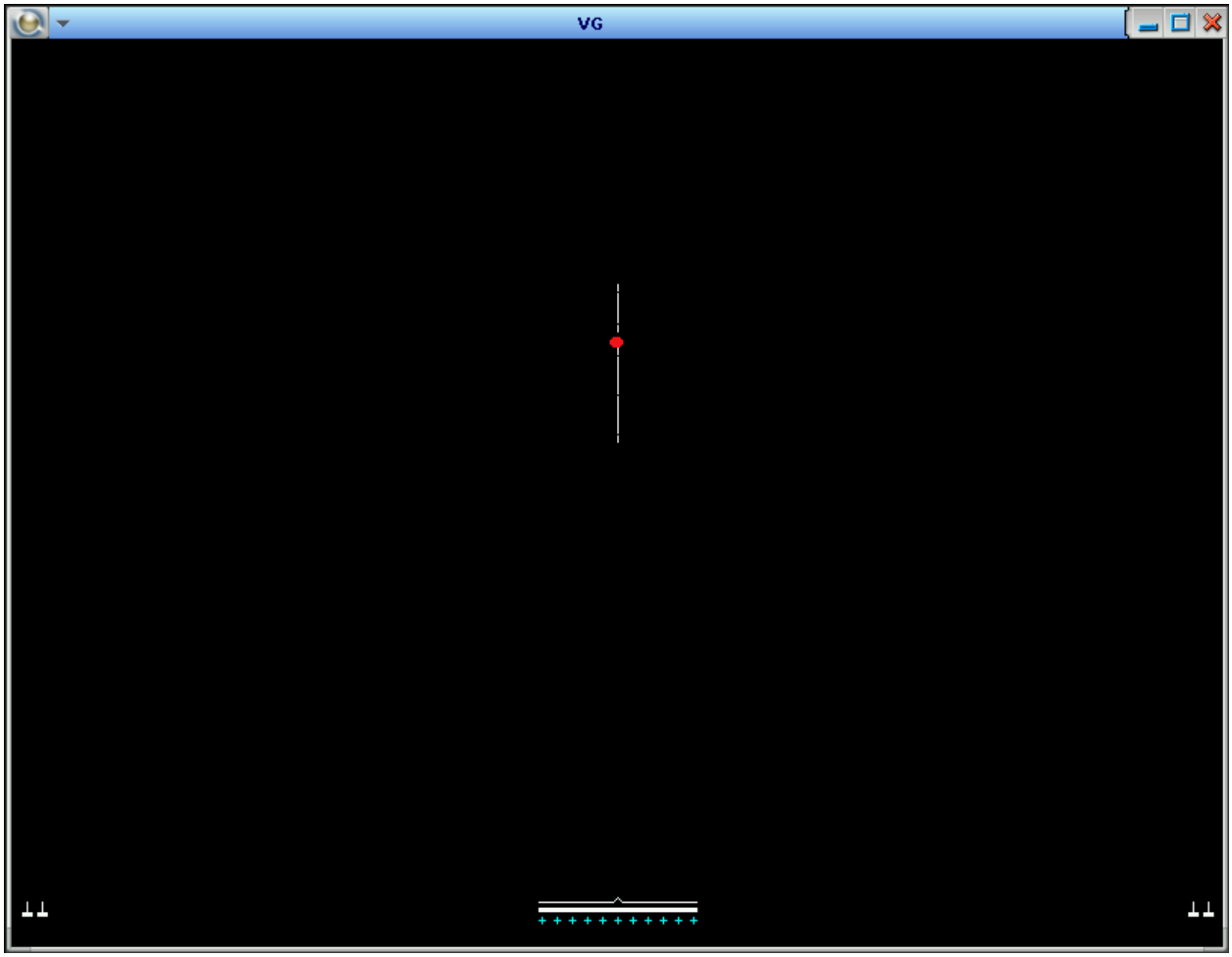


```
#include <sys/neutrino.h>
#include <unistd.h>
#include <vingraph.h>
#include <stdio.h>
#include "/root/labs/plates.h"

int main () {
    char c;
    StartGame (1);
    while (true) {
        c = InputChar();
        if (c == '0') {
            putreg(RG_RCMN, c-'0'); // 200x200
            putreg(RG_RCMC, RCMC_START);
            usleep(800000); //400ms ~ 100px
            putreg(RG_RCMC, RCMC_RIGHT);
            usleep(800000);
            putreg(RG_RCMC, RCMC_DOWN);
            usleep(800000);
            putreg(RG_RCMC, RCMC_LEFT);

            putreg(RG_RCMN, 1); // 500x200
            putreg(RG_RCMC, RCMC_START);
            putreg(RG_RCMC, RCMC_LEFT);
            usleep(1000000);
            putreg(RG_RCMC, RCMC_UP);
            usleep(800000);
            putreg(RG_RCMC, RCMC_RIGHT);
            usleep(2000000);
            putreg(RG_RCMC, RCMC_DOWN);
            usleep(800000);
            putreg(RG_RCMC, RCMC_LEFT);
        }
    }
    EndGame();
    return 0;
}
```

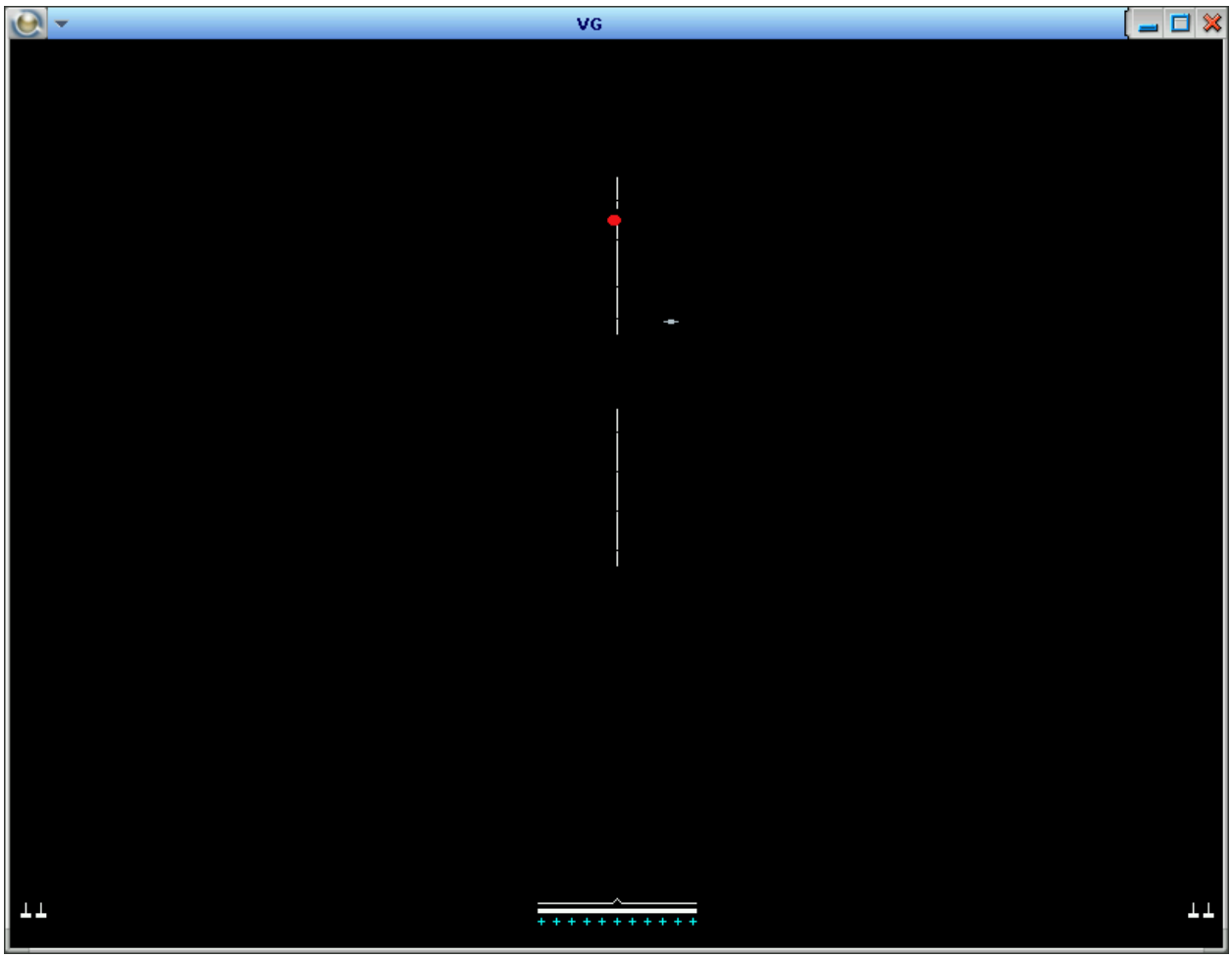
2. Написать программу, сбивающую одну тарелку с помощью ракеты (тарелка движется слева направо).



```
#include <sys/neutrino.h>
#include <unistd.h>
#include <vingraph.h>
#include <stdio.h>
#include "/root/labs/plates.h"
#include <time.h>
#include <math.h>
#include <iostream>

int main () {
   char c;
   StartGame (1);
   struct timespec start, stop;
   double whenToFire = 0;
   while (true) {
      usleep(1);
      if(getreg(RG_LOCN) == 1 && getreg(RG_LOCW) == 3) { // left-right
         clock_gettime(CLOCK_REALTIME, &start);
         while (true) {
            usleep(1);
            if (getreg(RG_LOCN) == 2) {
               clock_gettime(CLOCK_REALTIME, &stop);
               break;
            }
         }
         double usec = (stop.tv_sec - start.tv_sec) * 1000000 + (stop.tv_nsec - start.tv_nsec) / 1000.0;
         double targetSpeed = 10  / usec;
         double targetCenterTime = 380.0 / targetSpeed;
         double timeToFly = ((570 - getreg(RG_LOCY) ) / 100.0)*1000000;
         whenToFire = targetCenterTime - timeToFly - 300000;
         usleep(whenToFire);
         for (int i = 0; i < 20; i++) {usleep(50000);putreg (RG_GUNS, GUNS_SHOOT);}
      }
   }
   EndGame();
   return 0;
}
```

3. Написать программу, сбивающую несколько тарелок с помощью ракет (тарелки движутся в разных направлениях).
4. Написать программу, сбивающую медленные тарелки ракетами, а быстрые - управляемыми снарядами.


VG


VG

```
#include <sys/neutrino.h>
#include <unistd.h>
#include <vingraph.h>
#include <stdio.h>
#include "/root/labs/plates.h"
#include <time.h>
#include <math.h>
#include <iostream>
#include <pthread.h>
#include <sys/mman.h>

void *leftright(void *args);
void *rightleft(void *args);
static int *ammo;

int main ()
{
    ammo = static_cast<int*>(mmap(NULL, sizeof *ammo, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1,0));
    *ammo = 0;
    pthread_t lr, rl;
    StartGame (2);
    pthread_create(&lr, NULL, &leftright, NULL);
    pthread_create(&rl, NULL, &rightleft, NULL);
    pthread_join(lr, 0);
    pthread_join(rl, 0);
    EndGame();
}

void *rightleft(void *args)
{
double whenToFire = 0;
    struct timespec start, stop;
    while(true)
    {
        usleep(1);
        if(getreg(RG_LOCN) == 4 && getreg(RG_LOCW) == 3)
        {
            clock_gettime(CLOCK_REALTIME, &start);
            while (true)
            {
                usleep(1);
                if (getreg(RG_LOCN) == 3) {
                    clock_gettime(CLOCK_REALTIME, &stop);
                    break;
                }
            }
            long nsecs = stop.tv_nsec - start.tv_nsec;
            long secs = stop.tv_sec - start.tv_sec;
```

```
            long nsecs = stop.tv_nsec - start.tv_nsec;
            long secs = stop.tv_sec - start.tv_sec;
            if (secs > 0 && nsecs < 0)
            {
               nsecs += 1000000000;
               secs--;
            }
            else if (secs < 0 && nsecs > 0)
            {
               nsecs -= 1000000000;
               secs++;
            }
            double usec = (secs * 1000000000 + nsecs) / 1000.0;
            int HEIGHT = getreg(RG_LOCY);
            double targetSpeed = 10. / usec;
            double targetCenterTime = 380.0 / targetSpeed;
            double timeToFly = ((570 - getreg(RG_LOCY) ) / 100.0)*1000000;
            whenToFire = targetCenterTime - timeToFly - 300000;
            if (usec <= 60000 || whenToFire > 11000000 || whenToFire < 0.3)
            {
               std::cout << "Guided shell launch" << std::endl;
               if (*ammo <= 9)
               {
                  int timeOfFlight = 4000*(570-HEIGHT);
                  putreg(RG_RCMN, *ammo);
                  putreg(RG_RCMC, RCMC_START);
                  usleep(timeOfFlight);
                  putreg(RG_RCMC, RCMC_RIGHT);
                  *ammo += 1;
               }
            }
            else
            {
               usleep(whenToFire);
               for (int i = 0; i < 20; i++) {usleep(50000);putreg (RG_GUNS, GUNS_SHOOT);}
            }
         }
      }
   }

void *leftright(void *args)
{
   double whenToFire = 0;
   struct timespec start, stop;
   while(true)
   {
      usleep(1);
      if(getreg(RG_LOCN) == 1 && getreg(RG_LOCW) == 3)
```

```
    usleep(1);
    if(getreg(RG_LOCN) == 1 && getreg(RG_LOCW) == 3) {
        start.tv_sec = 0;
        start.tv_nsec = 0;
        stop.tv_sec = 0;
        stop.tv_nsec = 0;
        clock_gettime(CLOCK_REALTIME, &start);
        while (true) {
            usleep(1);
            if (getreg(RG_LOCN) == 2) {
                clock_gettime(CLOCK_REALTIME, &stop);
                break;
            }
        }
        long nsecs = stop.tv_nsec - start.tv_nsec;
        long secs = stop.tv_sec - start.tv_sec;
        if (secs > 0 && nsecs < 0) {
            nsecs += 1000000000;
            secs--;
        }
        else if (secs < 0 && nsecs > 0) {
            nsecs -= 1000000000;
            secs++;
        }
        int HEIGHT = getreg(RG_LOCY);
        double usec = (secs * 1000000000 + nsecs) / 1000.0;
        double targetSpeed = 10.  / usec;
        double targetCenterTime = 380.0 / targetSpeed;
        double timeToFly = ((570. - getreg(RG_LOCY) ) / 100.0)*1000000.;
        whenToFire = targetCenterTime - timeToFly - 300000;
        if (usec <= 60000 || whenToFire > 11000000 || whenToFire < 0.3){
            std::cout << "Guided shell launch" << std::endl;
            if (*ammo <= 9) {
                int timeOfFlight = 4000*(570-HEIGHT);// 4000 usec = 1px
                putreg(RG_RCMN, *ammo);
                putreg(RG_RCMC, RCMC_START);
                usleep(timeOfFlight);
                putreg(RG_RCMC, RCMC_LEFT);
                *ammo += 1;
            }
        }
        else {
            usleep(whenToFire);
            for (int i = 0; i < 20; i++) {usleep(50000);putreg (RG_GUNS, GUNS_SHOOT);}
        }
    }
  }
}
```