|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Московский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных Технологий

Кафедра математического обеспечения и стандартизации информационных технологий

**ОТЧЕТ ПО ПРОЕКТУ «Наш пинг-понг»**

по дисциплине

«СиП инженерия»

Учебная группа: ИВБО-02-16

Студен: Бхаттарай Й.Р.

Преподаватель: Миронов А. Н.

Москва 2018

**Оглавление**

[Постановка задачи 3](#_Toc530938177)

[Выполнение работы 3](#_Toc530938178)

[Исходный код (листинг) программ с комментариями 5](#_Toc530938179)

[Выводы 6](#_Toc530938180)

# Постановка задачи

Реализовать 2 функции:

1. Расчет следующей координаты шарика в режиме реального времени.

На вход функция получает текущие координаты, время предыдущего изменения, текущее время, скорость шарика, угол полета шарика.

Возвращает функция вычисленную координату шарика.

1. Расчет угла полета шарика после встречи с препятствием.

На вход функция получает текущий угол, угол препятствия.

Возвращает функция новый угол движения.

Реализация на языке С++.

# Выполнение работы

Для удобства программирования был создан заголовочный файл info.h имеющий следующее содержание:

#pragma once

#include <math.h>

#include <cstdlib>

#include <ctime>

struct coord {

float x;

float y;

};

Затем была написана функция расчета изменения угла. Предполагается, что при отражении от нижней и верхней границы игрового поля угол будет просто отражаться по оси Y, поэтому формула имеет вид (2π - текущий угол). Расчет происходит в радианах. Переходы угла между четвертями тригонометрической окружности можно описать как (рис.1):

* 1->4
* 2->3
* 3->2
* 4->1

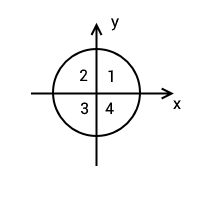


Рис.1. Тригонометрическая окружность

Предполагается, что шар движется под углом какой-то четверти.

В случае отражения от платформ существует два случая: угол движения лежит в 1 и 2 четверти или 3 и 4. В первом случае функция имеет вид (π - текущий угол), во втором — (3π - текущий угол). Так же добавляется малое случайное изменение угла, чтобы шар не двигался только влево-вправо. Переходы угла между четвертями тригонометрической окружности можно описать как (рис.1):

* 1->2
* 2->1
* 3->4
* 4->3

Таблица 1. Переменные функции angle.cpp

|  |  |  |
| --- | --- | --- |
| Переменные | Тип | Пояснения |
| angFw | float | Текущий угол движения. |
| angOfObst | float | Угол препятствия (для отличия границ от платформ). |
| angBack | float | Угол отражения. |

Функция следующей координаты разбивает движение шара на перемещение по y и по x, используя синус и косинус соответственно.

Таблица 2. Переменные функции coordinate.cpp

|  |  |  |
| --- | --- | --- |
| Переменные | Тип | Пояснения |
| coordNow | coord | Текущая координата. |
| prevChangeTime | int | Время предыдущего изменения. |
| timeNow | int | Время в данный момент. |
| speed | int | Скорость движения шара. |
| angle | float | Угол движения шара. |
| next | coord | Новые координаты шара. |
| t | int | Время изменения. |

# Исходный код (листинг) программ с комментариями

angle.cpp

//расчет изменения угла

#include "info.h"

float angleChange(float angFw, float angOfObst)

{

float angBack;

srand(time(NULL)); //для случайного изменения угла при столкновении с платформой

if (angOfObst == 0) { //при столкновении с верхней/нижней границей поля

angBack = 2\*PI - angFw;

}

else { //отражение от платформ, угол случайно немного меняется

case ((angFw >= 0) && (angFw<=(PI))): //движение вверх

angBack = PI - angFw + ((rand() % 1200 + 1000)/10000);

break;

case ((angFw >= PI) && (angFw<=(2\*PI))): //движение вниз

angBack = 3\*PI - angFw + ((rand() % 1200 + 1000)/10000);

break;

}

return angBack; //возврат нового угла движения

}

coordinate.cpp

//расчет следующей координаты

#include "info.h"

struct coord nextCoordinate(coord coordNow, int prevChangeTime, int timeNow, int speed, float angle)

{

coord next;

int t = timeNow - prevChangeTime; //время изменения

next.y = coordNow.y + (t\*speed\*(sin(angle))); //измененная y-координата

next.x = coordNow.x + (t\*speed\*(cos(angle))); //измененная х-координата

return next; //возвращаем новые координаты

}

# Статистика с github

