KTH Stockholm
CSC :: CST
Introduction to Visualization and Computer Graphics, Spring 2017, DH2320
Tino Weinkauf, Himangshu Saikia, Wiebke Köpp

Homework assignment No. 05

Due February 24, 2017

In this task you will implement Chaikin's corner cutting algorithm and Bézier curves in order to make two tanks (which are represented by spheres) follow a curve and shoot at each other. The initial setting is shown in Fig.1.

The two tanks follow line segments across a terrain. Upon pressing the space bar the tanks halt and a shot originates from a corner of the terrain and hits one of the tanks. You should get a similar result to Fig. 1 after following the instructions for **Assignment 0** (Environment Setup), setting the project '*task*' as the 'start up' project/target and running the program. You will only need to edit the file `task.h`.

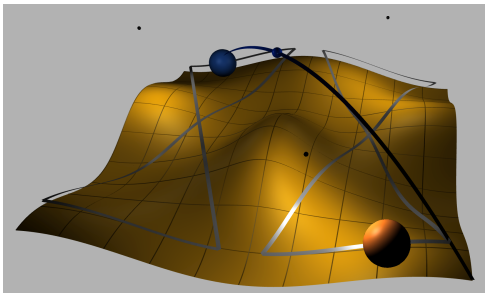

Figure 1: Initial setting

Task 5.1: Chaikin's Corner Cutting **10 P**

Implement Chaikin's corner cutting algorithm, so that both tanks follow a smooth curve. The result should look similar to the scene in Fig. 2. You need to edit the function `void Chaikin(...)` which takes the parameters below.

- `const std::vector< ogl::Vec2f >& ControlPolygon`: points defining the line segments to be subdivided
- `const size_t MinNumDesiredPoints`: minimum number of points the resulting curve should contain
- `std::vector< ogl::Vec2f >& Curve`: points of the resulting curve after applying corner cutting

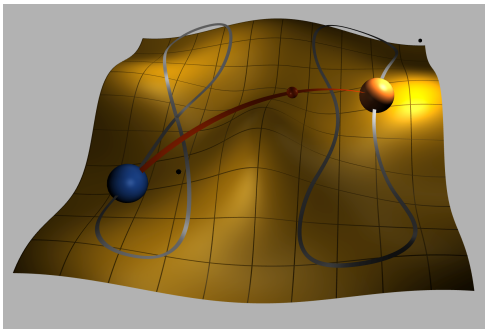

Figure 2: Result of Task 1

Task 5.2: Bézier Curves: Quadratic Curve **10 P**

After completing this task the tanks interact by shooting each other. We assume negligible air resistance and tanks capable of aiming perfectly. That means a projectile travels from one of the tanks to the other along a parabolic path, i.e., a quadratic Bézier curve (see Fig. 3).

You will need to adapt the functions `void DefineBallisticParabolaControlPolygon(...)` and `void Bezier(...)`. The former defines the control polygon for a ballistic curve given the positions of both the shooting and the targeted tank while the latter computes a given number of points on that curve.

Your control polygon should be defined such that the trajectory of the projectile starts at the shooting tank and ends at the targeted tank. The curve in between is a quadratic Bézier curve. You are free to choose either the de Casteljau algorithm or use Bernstein basis polynomials in order to compute the desired number of points on the curve.

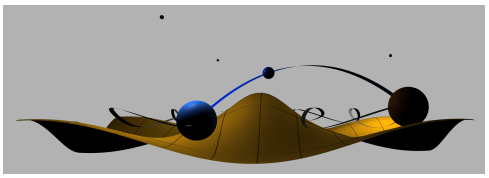

Figure 3: Result of Task 2

Task 5.3: Bézier Curves: Arbitrary number of control points (Bonus) **10 BP**

Adapt the function `void Bezier(...)` so that it is able to compute higher-order curves as well.