Comuptergrafik

Universität Bern

Herbst 2010

Assignment 6

- This time you have 3 weeks!
  - Turn-in is december 20

Assignment 6

- Exercise 1:

Create a rotational body from beziercurves using appropriate vertex positions, normal vectors, texture coordinates and an index array

- Exercise 2:

Create a scene composed of several rotaitonal bodies

- Exercise 3:

Implement the loop subdivion algorithm using a (given) winged edge structure

- BONUS ASSIGNMENT:

More about this later…

Exercise 1

1. Define control points on a plane

2. Approximate the curve for arbitrary positions

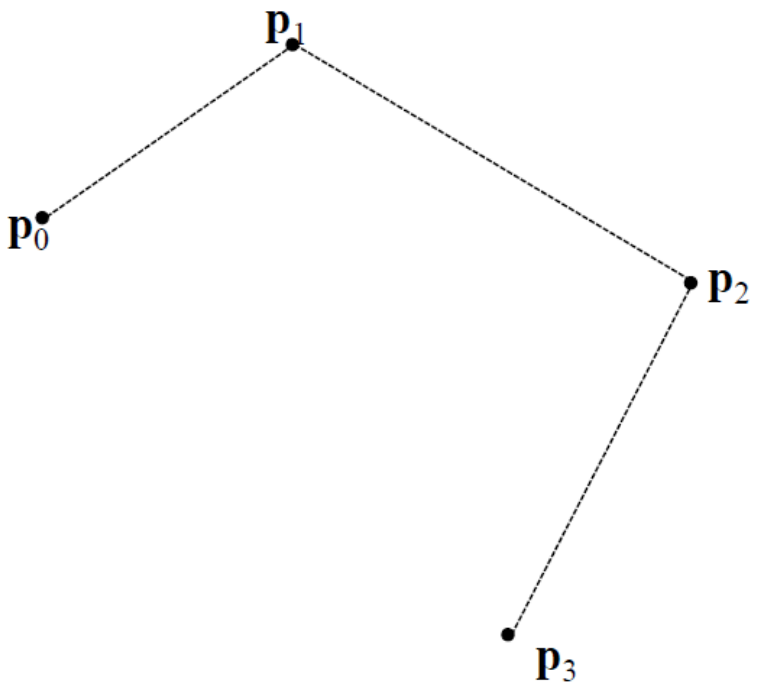
3. Rotate the resulting points around an axis

4. Connect the points to a triangle mesh

5. Compute the normals and texture coordinates of each vertex

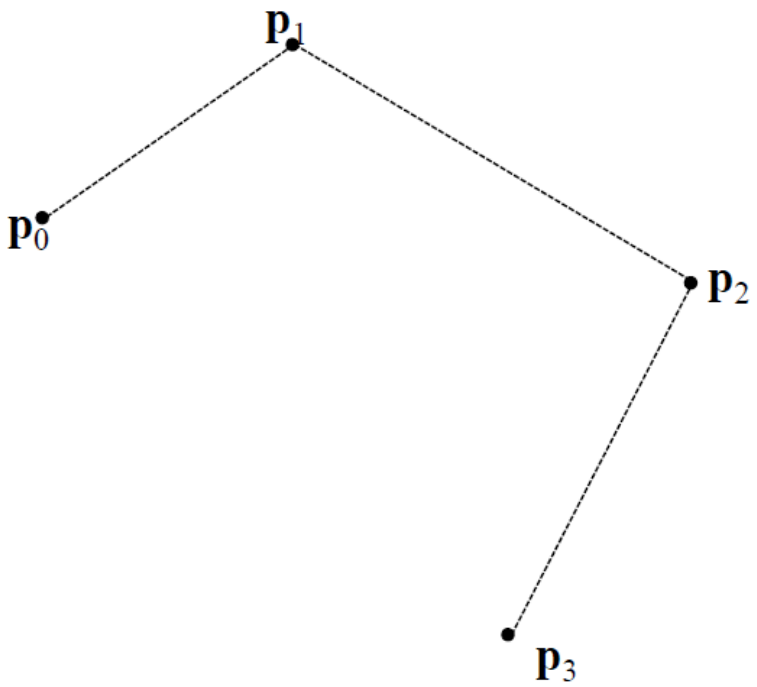
Exercise 1

1. Define control points on a plane



Exercise 1

1. Define control points on a plane



n cubic bezier segments require

(n-1)*3+4 control points!

Exercise 1

2. Approximate the curve for arbitrary positions

Exercise 1

2. Approximate the curve for arbitrary positions

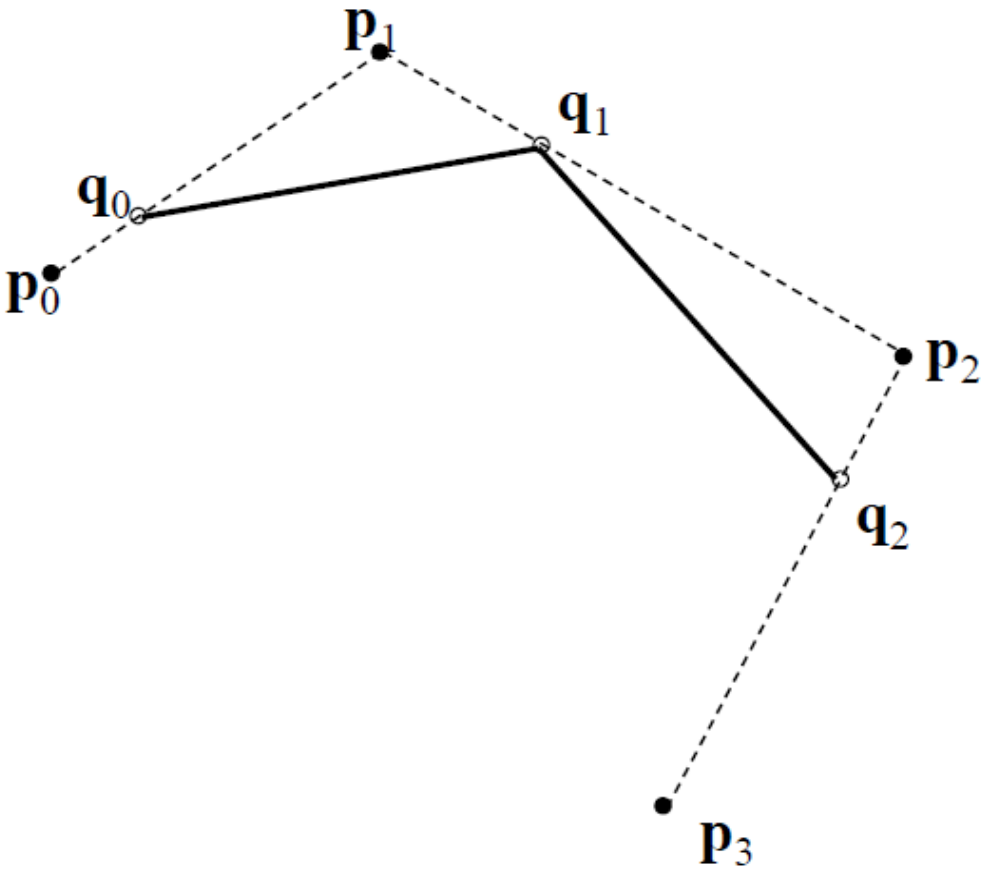
Use the **deCasteljau** algorithm!

Exercise 1

2. Approximate the curve for arbitrary positions

Use the **deCasteljau** algorithm!

$$\mathbf{q}_0(t) = Lerp(t, \mathbf{p}_0, \mathbf{p}_1)$$
$$\mathbf{q}_1(t) = Lerp(t, \mathbf{p}_1, \mathbf{p}_2)$$
$$\mathbf{q}_2(t) = Lerp(t, \mathbf{p}_2, \mathbf{p}_3)$$



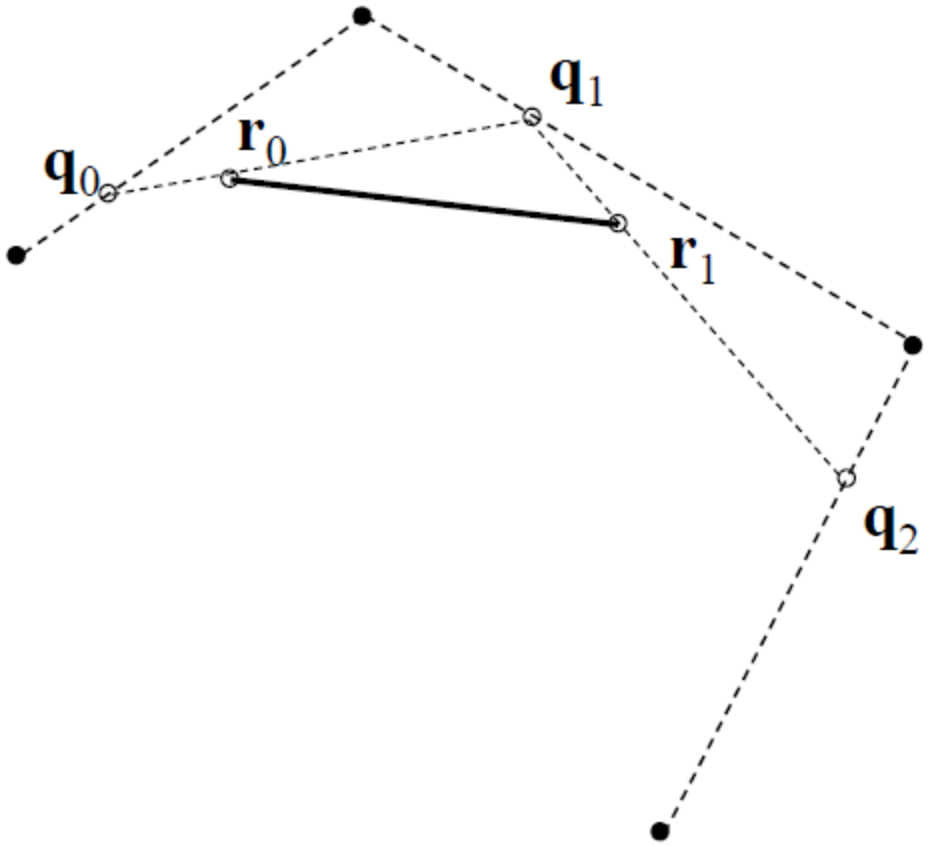
Exercise 1

2. Approximate the curve for arbitrary positions

Use the **deCasteljau** algorithm!

$$\mathbf{r}_0(t) = Lerp\left(t, \mathbf{q}_0(t), \mathbf{q}_1(t)\right)$$

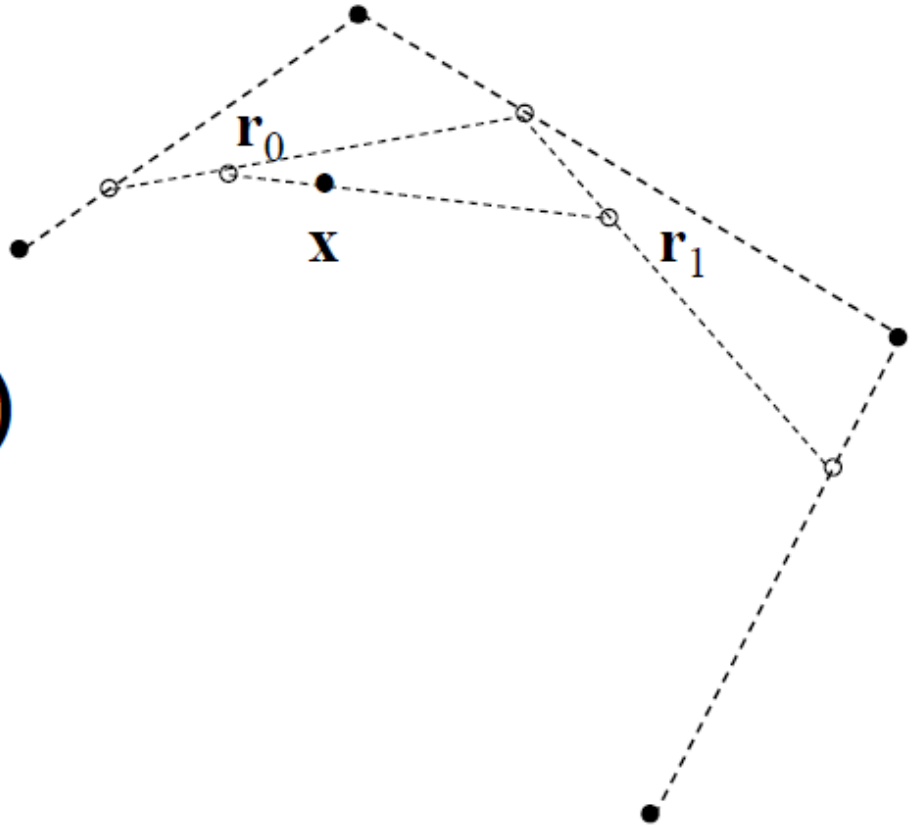
$$\mathbf{r}_1(t) = Lerp\left(t, \mathbf{q}_1(t), \mathbf{q}_2(t)\right)$$



Exercise 1

2. Approximate the curve for arbitrary positions

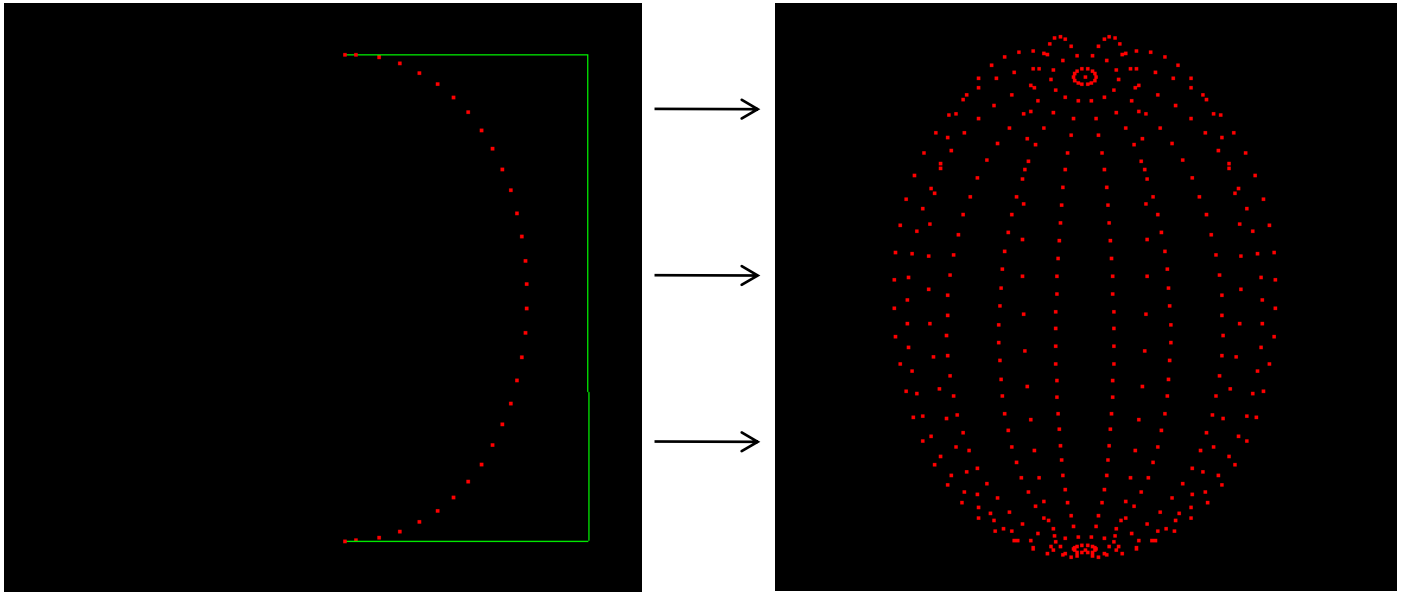
Use the **deCasteljau** algorithm!



$$\mathbf{x}(t) = Lerp\left(t, \mathbf{r}_0(t), \mathbf{r}_1(t)\right)$$

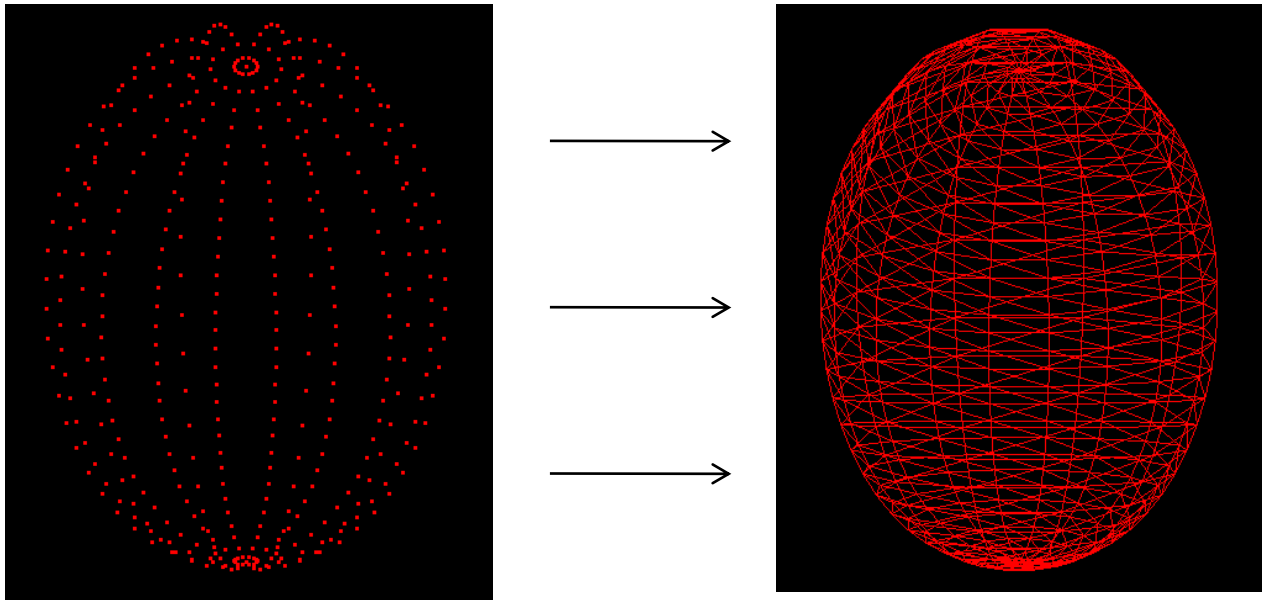
Exercise 1

3. Rotate the resulting points around an axis



Exercise 1

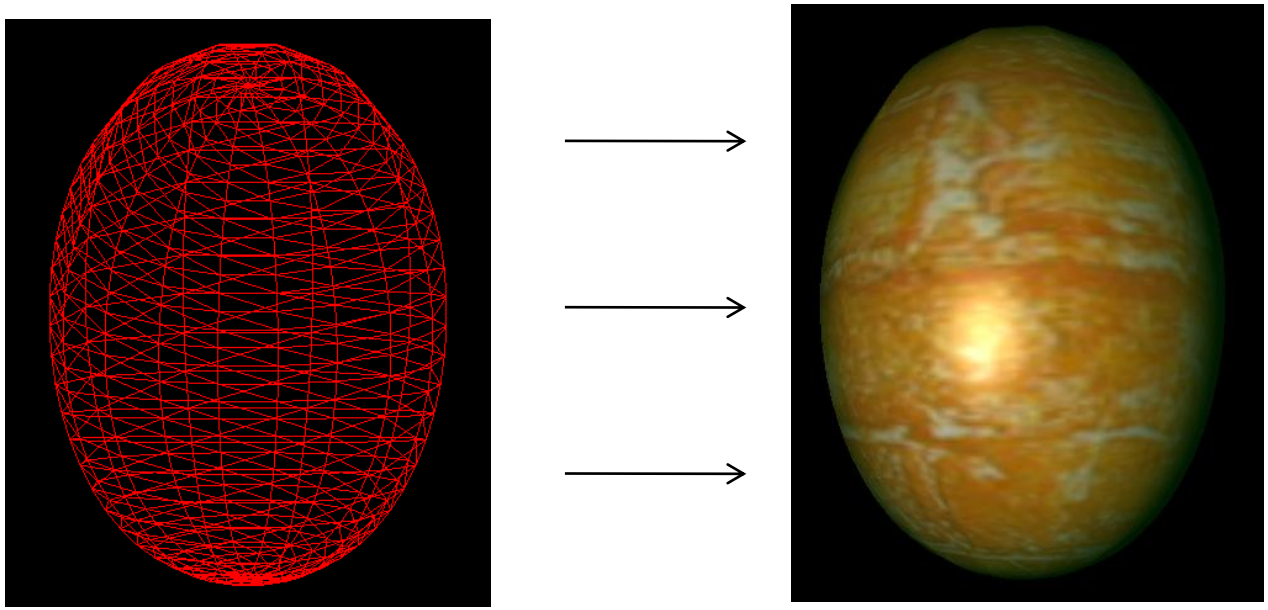
4. Connect the points to a triangle mesh



Create an index array (similar to assignment 1)

Exercise 1

5. Compute the normals and texture coordinates of each vertex



Exercise 1

5. Compute the normals and texture coordinates of each vertex

Compute normals:

- For each vertex on the 2D bezier plane:
  - Compute tangent (x,y,0)
  - Normal is then (-y,x,0)
- Rotate the normal together with the vertex on the 2D bezier plane

Exercise 2

Implement the loop subdiviosion algorithm.

Exercise 2

Implement the loop subdiviosion algorithm.

•The given code already implements a conversion of vertex and face tables to **winged edge structures**!

Exercise 2

Implement the loop subdiviosion algorithm.

- The given code already implements a conversion of vertex and face tables to **winged edge structures**!

- Methods to find adjacent faces, vertices and edges using the winged edge structure are already implemented.

Exercise 2

Implement the loop subdiviosion algorithm.

- The given code already implements a conversion of vertex and face tables to **winged edge structures**!

- Methods to find adjacent faces, vertices and edges using the winged edge structure are already implemented.

- All you need to do is implement the loop algorithm itself!

Exercise 2

Sidenote:

The given code only works on closed triangle meshes without borders.

- Closed means the mesh is not allowed to have holes in it.
- Without borders means that **each edge** in the mesh **must have <u>exactly two</u> neighbor faces**.

Exercise 2

Sidenote:

The given code only works on closed triangle meshes without borders.

- Closed means the mesh is not allowed to have holes in it.
- Without borders means that **each edge** in the mesh **must have <u>exactly two</u> neighbor faces**.

This means most of the meshes you used until now won‘t work!

Therefore, use the new meshes we provide to test your algorithm!

Exercise 2

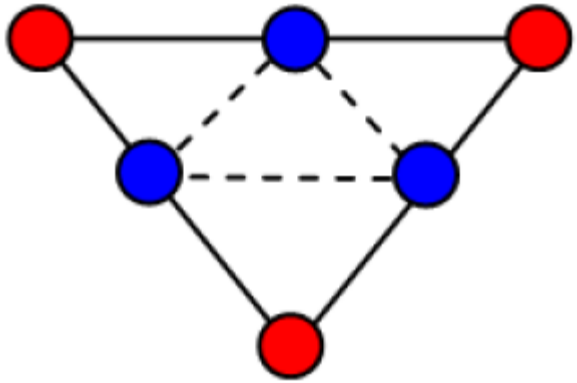
Loop sufrace subdivision (reminder):

2 Steps:

- Subdivision
- Smoothing

Exercise 2

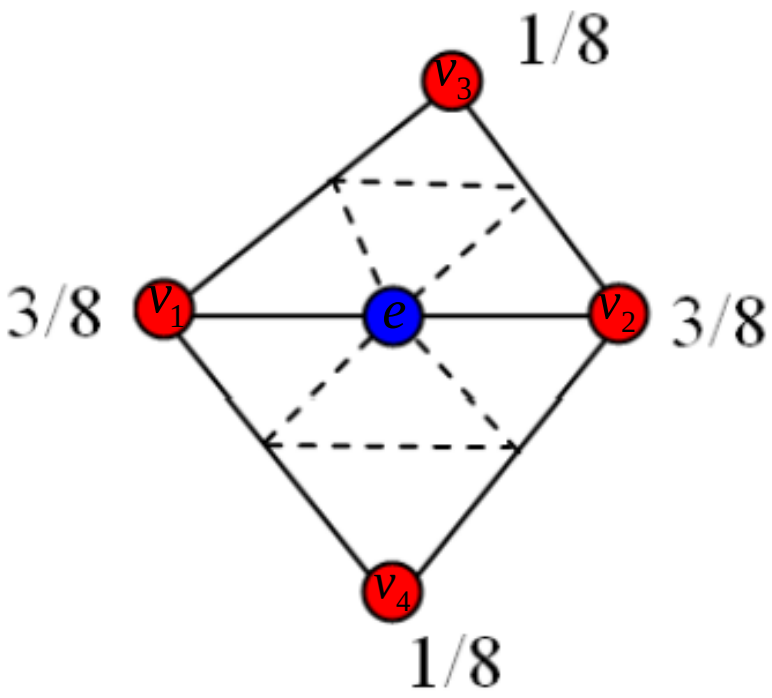
- Subdivision:
  - Split each triangle into four



Exercise 2

- Smoothing:
  - For **new** vertices:

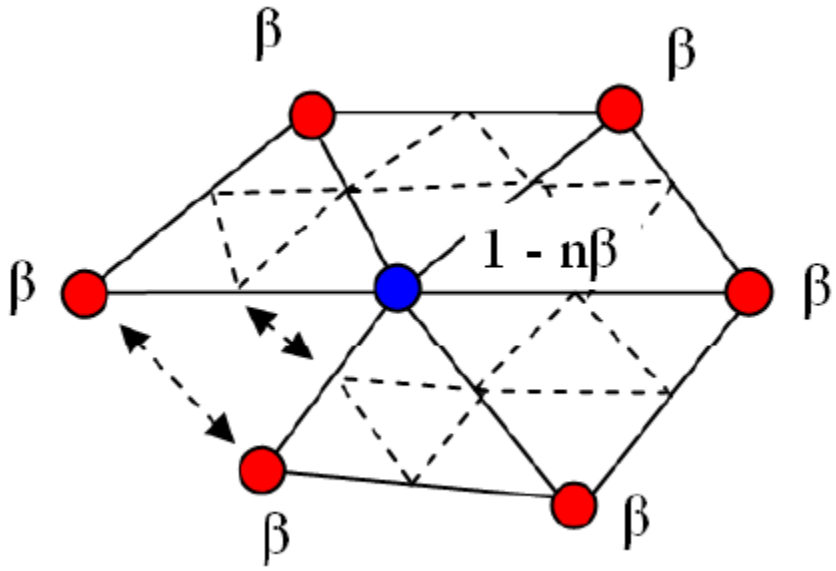
$$e = \frac{3v_1 + 3v_2 + v_3 + v_4}{8}$$



Exercise 2

- Smoothing:
  - For **old** vertices:

$$v' = (1 - n\beta)v + \beta \sum_{k=1}^{n} v_k$$

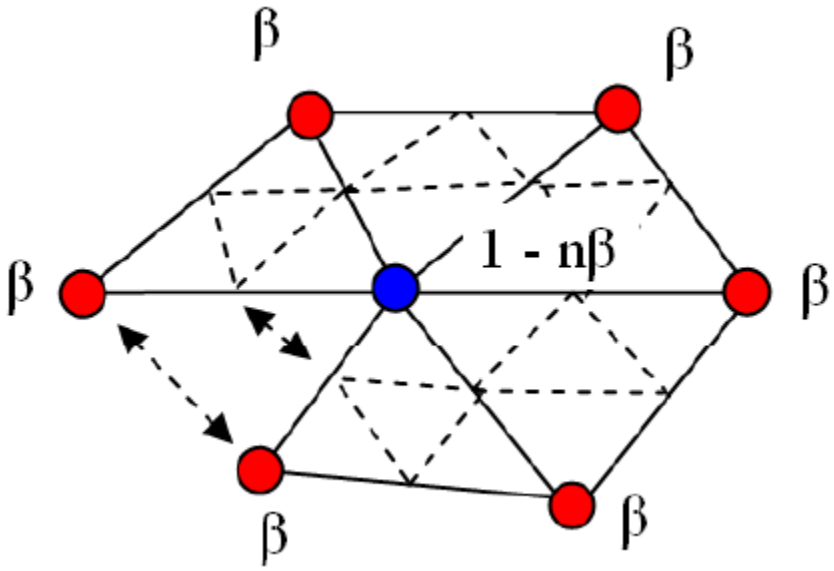


Exercise 2

- Smoothing:
  - For **old** vertices:

$$v' = (1 - n\beta)v + \beta \sum_{k=1}^{n} v_k$$

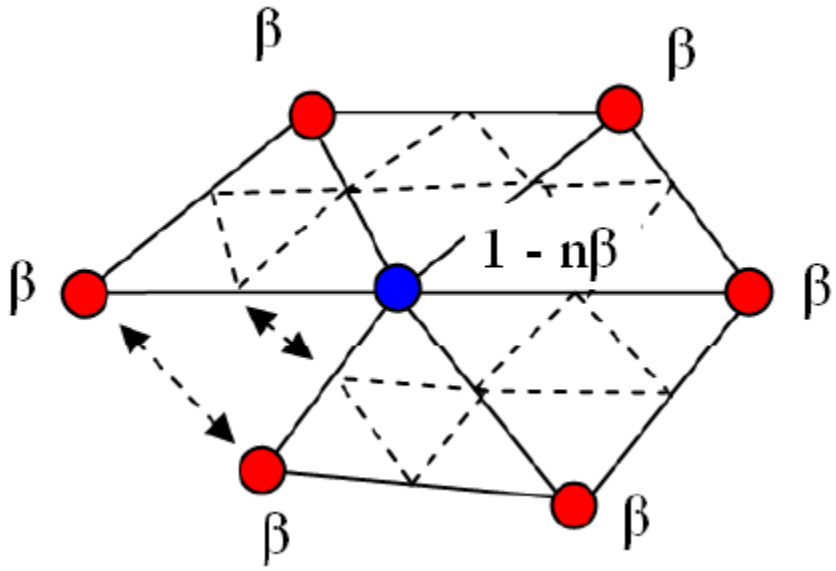
Number of neighbor vertices



Exercise 2

- Smoothing:
  - For **old** vertices:

$$v' = (1 - n\beta)v + \beta \sum_{k=1}^{n} v_k$$



$$\beta = \frac{3}{8n} \quad \text{if} \quad n > 3$$

Where

$$\beta = \frac{3}{16} \quad \text{if} \quad n = 3$$

Exercise 3

Model a scene composed of at least **3 <u>different</u> rotational bodies or subdivided surfaces**.

Exercise 3

Model a scene composed of at least **3 <u>different</u> rotational bodies or subdivided surfaces**.

•Each object should have its own material, textures and color.

Exercise 3

Model a scene composed of at least **3 <u>different</u> rotational bodies or subdivided surfaces**.

- Each object should have its own material, textures and color.

- Some ideas:
  - A round table with some objects on it (vases, candles, plates etc.)
  - A basket with some fruits in it.
  - A checker board with some chess pieces on it
  - etc.

Bonus Assignment

- Bonus assignment won‘t give you any points…

Bonus Assignment

- Bonus assignment won‘t give you any points…

… but depending on how good you do it, you receive a **bonus of up to 0.5 on the courses final grade!**

Bonus Assignment

- You need to implement at least one of the following algorithms:

- Shadow Mapping
- Bump Mapping
- Refraction/Reflection with Environment Maps
- Irradiance Environment Maps
- Ambient Occlusion
- Catmull-Clark surface subdivision

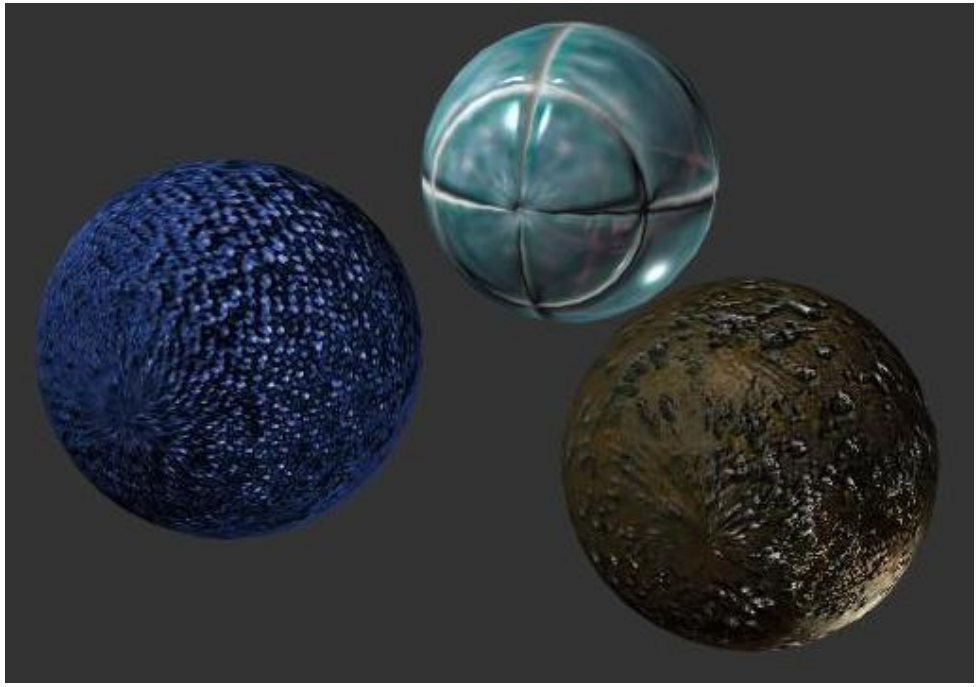
Bonus Assignment

Shadow Mapping

- Use Percentage Closer Filtering
- Lighsource should be interactively movable

Bonus Assignment

Bump Mapping



- Lightsource should be interactively movable
- use xNormals to compute the tangent vectors (http://www.xnormal.net/1.aspx)

Bonus Assignment

Refraction and reflection with Environment Maps

•Use the schlick approximation for the Fresnel equation

Bonus Assignment

Irradiance Environment Maps

•Use HDRShop to generate the irradiance map (http://www.hdrshop.com/)

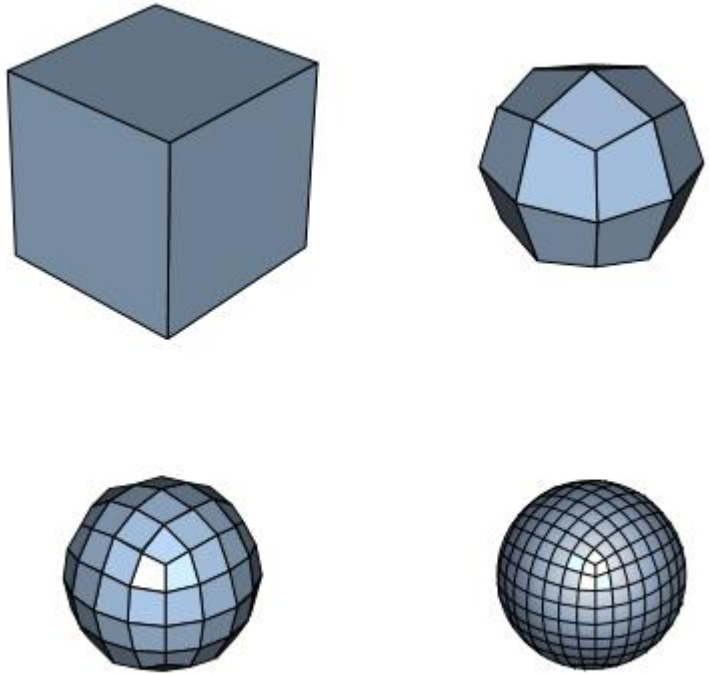
Bonus Assignment

Ambient Occlusion

- You can also use xNormals for this

Bonus Assignment

Catmull-Clark Surface subdivision



•Must be applicable on arbitrary closed mesh (not just triangle meshes)

Bonus Assignment

Implementation of the algorithm should be demonstrated by using a nice scene.

Bonus Assignment

Implementation of the algorithm should be demonstrated by using a nice scene.

We rate the technical difficulty of the implemented algorithm(s) as well as the asthetic impression of your final scene.

Questions?