Project Option A: A procedurally generated 3D animation

Your task is to write a procedurally generated (*non-interactive*) 3D real-time animation of approximately 20-30 seconds duration, using the OpenGL API, which must have the following features:

1. procedurally generated geometry.
2. procedurally generated textures.
3. procedurally animated camera and objects.
4. at least 1 cut, a maxium of 3 cuts during the animation.
5. use one of the following themes:
    - The Battle of Helm's Deep (*Lord of the Rings: The Two Towers*)
    - The Shadow War (*Babylon5*)
    - Bug attack on Klendathu (*Starship Troopers*)
    - The battle of Wolf 359 (*Star Trek: The Next Generation*)
    - The Red Shoes (*Andersen's Fairy Tales*)
    - The Steadfast Tin Soldier (*Andersen's Fairy Tales*)
    - Abstract Animation (30 seconds)
    - Theme X (*your choice – after consultation with your lecturer*)

Extra features or other innovative content in terms of graphics (*design*) will be part of your overall mark.

You have to provide

a) One executable file
b) All sourcecode files (*commented*) needed for compiling the project
c) Project Report (*up to ca. 6 A4 pages*)
d) HTML User Manual for your application (*max. 2 A4 pages when printed*)

References & Resources:

3D demos	- http://www.scene.org/
fr-08 .the.product demo	- http://www.xyzw.de/x002.html
fr-25 the popular demo	- http://www.xyzw.de/x011.html

Project Option B: A "painting by numbers" image converter

Your task is to write a "painting by numbers" image converter that creates a "painting by numbers" template picture from a given source image. Use a programming environment of your choice to create a tool that is capable of the following:

1. Loading image data.
2. Conversion of the image to a painting-by numbers image (*noise reduction, colour reduction – see 3., outlines*) using image processing techniques.
3. Selection of a colour palette* for commercially available paints (*can be found on most manufacturer's homepages*) and numbering of picture sections according to this palette.
4. A graphical or a menu-driven (*curses*) user interface.
5. Saving "painting by numbers" images in addition to a key (*list*) of colours required as well as saving of the intermediate images as (*optionally selectable*) debug-output.

Extra features or other innovative content in terms of graphics or interface design will be part of your overall mark.

You have to provide

a) The executable tool
b) All source code files (*commented*) needed for compiling the project
c) Project Report (*up to ca. 6 A4 pages*)
d) HTML User Manual for your application (*max. 4 A4 pages when printed*)
e) A sample project

References & Resources:

file format descriptions can be found at: http://www.wotsit.org
various image processing articles at Generation5:

- http://www.generation5.org/content/2004/noiseIntro.asp
- http://www.generation5.org/content/2002/im01.asp
- http://www.generation5.org/content/2002/convolution.asp
- http://www.generation5.org/content/2003/segmentation.asp
- http://www.generation5.org/content/2001/im00.asp
- http://www.generation5.org/content/2004/histogramEqualization.asp

Painting by Numbers" examples - http://www.craftbynumbers.com/painting/examples/

*colour charts example: http://www.daler-rowney.com/prod_catalogue/colourchart.asp
palette-based quantization : http://msdn2.microsoft.com/en-us/library/aa479306.aspx

Project Option C: A "sheep control" language

Your task is to write a compiler for a simple behaviour definition language for sheep on "The Meadow", which is capable of the following:

1. Interpreting "Sheep Behaviour" Programs
   - executing programs in a graphical environment using the OpenGL API (*possibly employing the c-sheep companion library*)

   with the following language features:
   - moving the sheep forward/backwards
   - rotating the sheep left or right by 90 degrees
   - sensing obstacles and objects in the immediate environment
   - declare variables for holding numeric values
   - arithmetic operations for use with variables and numeric values
   - declare procedures/sub-routines with up to 2 parameters
   - call/execute procedures/sub-routines (*recursively*)

2. Cross-compiling your language to C-Sheep programs, compilable by the C-Sheep compiler or alternatively compiling your language into C-Sheep bytecode.

Extra features or other innovative content in terms of graphics or interface design will be part of your overall mark.

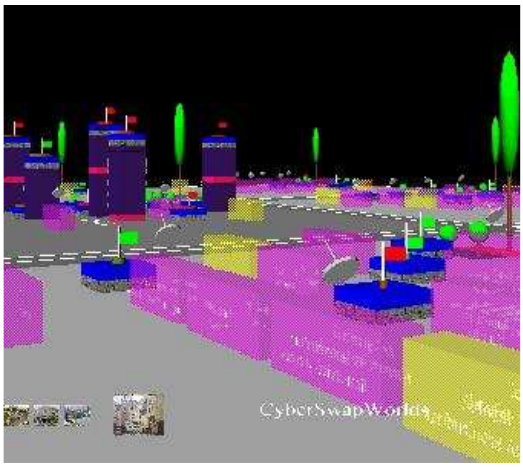
You have to provide

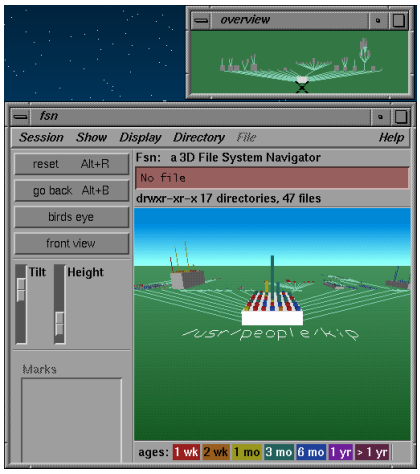
a) One executable file
b) All sourcecode files (*commented*) needed for compiling the project
c) Project Report (*up to ca. 6 A4 pages*)
d) HTML User Manual for your application (*max. 4 A4 pages when printed*)
e) a sample program

References & Resources

Languages & Compilers (*by Peter Comninos*)
Compiler Construction - http://www.oberon.ethz.ch/WirthPubl/CBEAll.pdf
C-Sheep Website: http://nccastaff.bournemouth.ac.uk/eanderson/C-Sheep/

Project Option D: A graphical system shell





Your task is to write a graphical system shell (*including a graphical file system navigator*) using the OpenGL API, which is capable of the following:

1. selecting, opening and listing of directories
2. executing programs and shell-scripts
3. graphically distinct representations of files, directories & links (including access permissions)
4. allowing keyboard/text IO through an optional HUD (Head-Up Display)
5. being operated by mouse movements and mouse clicks (*keyboard navigational control optional*)

Extra features or other innovative content in terms of graphics or interface design as well as functionality will be part of your overall mark.

You have to provide

a) One executable file
b) All sourcecode files (*commented*) needed for compiling the project
c) Project Report (*up to ca. 6 A4 pages*)
d) HTML User Manual for your application (*max. 2 A4 pages when printed*)
e) 5 sample screen shots that were taken from the running shell

Resources

the OpenGL homepage is located at http://www.opengl.org

Project Option E: A simple Lemmings-like puzzle game

Your task is to write a 2D puzzle game (*using 3D graphics*) that is similar to the popular early 1990s game Lemmings (*description below* *) which must be playable (*but need not contain all of the original features*), using the OpenGL API, which must have the following features:

1. A custom file-format for levels. Levels can be interactively selected and loaded in by the user.
2. A side-view perspective viewport and a user interface which is operated by the keyboard.
3. Textured in-game 3D objects & environmental building blocks loaded from files of at least one of the following file formats:
   - DXF
   - Alias|Wavefront Object format
4. Simple animated player creatures.
5. At least 4 selectable behaviours/actions for the player creatures.

Extra features or other innovative content in terms of graphics (*design*) or additional in-game options will be part of your overall mark.

You have to provide

a) One executable file & all necessary resource files (*textures etc.*)
b) All source code files (*commented*) needed for compiling the project
c) Design & Implementation Documentation (*up to ca. 6 A4 pages*)
d) HTML User Manual for your application (*max. 2 A4 pages when printed*)
e) 3 sample levels for the game

*Lemmings

"The aim of the game is to guide a number of lemmings from the entrace to the exit in each level. The lemmings simply walk around and do nothing else themselves, so they'll run into the obstacles on the way.

However, the player can assign different tasks to a few of them, like digging, mining, blocking other lemmings and so on. By doing that, they can make the lemmings build a safe path for themselves to the exit. The aim is to save as many lemmings as possible in each level." (from: http://www.kallex.de/lemmings/games/)

References & Resources:

http://www.elizium.nu/scripts/lemmings/
http://en.wikipedia.org/wiki/Lemmings_(video_game)
http://pingus.seul.org/welcome.html

Project Option F: A simple side-scrolling arcade game

Your task is to write a 2D arcade game (*using 3D graphics*) that is similar to the popular late 1980s game Wings Of Fury (*WOF - description below* [*]) or alternatively a Jump&Run game (*similar to the popular "Super Mario Brothers" or "Sonic the Hedgehog"*) which must be playable (*but need not contain all of the original features*), using the OpenGL API, which must have the following features:

1. A custom file-format for levels. Levels can be interactively selected and loaded in by the user.
2. A side-view perspective viewport and a user interface which is operated by the keyboard.
3. Textured in-game 3D objects & environmental building blocks loaded from files of at least one of the following file formats:
   - DXF
   - Alias|Wavefront Object format
4. A simple animated player object.
5. Simple AI opponents.

Extra features or other innovative content in terms of graphics (*design*) or additional in-game options (*power-ups etc.)* will be part of your overall mark.

You have to provide

a) One executable file & all necessary resource files (*textures etc.*)
b) All source code files (*commented*) needed for compiling the project
c) Design & Implementation Documentation (*up to ca. 6 A4 pages*)
d) HTML User Manual for your application (*max. 2 A4 pages when printed*)
e) 1 sample level for the game (*spanning at least 4 horizontal screens*)

[*]WOF

"A semi-arcadish version of World War II carrier operations with the F6F Hellcat". The task of the player is to protect his aircraft-carrier during operations in the South Pacific by torpedoing Japanese carriers, shooting down Zero fighters and bombing Japanese airfields and outposts on the surrounding islands.

References & Resources:
http://www.classicgaming.com/amigareviews/wingsfur.htm#wingsoffuryzzap
http://www.bytethebullet.tk/

Project Option G: A simple top-down perspective tactical action game

Your task is to write a simple 2D top-down tactical action game (*using 3D graphics*) following similar gameplay concepts to popular games like UnrealTournament or "Return to Castle Wolfenstein". Elements of the game could take inspiration from the classic "Cannon Fodder" games (*see references*). The game should have a playable "capture the flag" game mode and should be created using the OpenGL API. The following features must be implemented:

1. A custom file-format for levels/maps. Levels can be interactively selected and loaded in by the user.
2. A user interface which is operated by the keyboard (*& possibly the mouse*).
3. textured in-game 3D objects & environmental building blocks loaded from files of at least one of the following file formats:
   - DXF
   - Alias|Wavefront Object format
4. Either a multi-player mode (*4 players minimum – possibly networked*) or simple (*but competent*) AI opponents & team-mates (*bots*).

Extra features or other innovative content in terms of graphics (*design*) or additional in-game options (*power-ups etc.)* will be part of your overall mark.

You have to provide

a) One executable file & all necessary resource files (*textures etc.*)
b) All sourcecode files (*commented*) needed for compiling the project
c) Project Report (*up to ca. 6 A4 pages*)
d) HTML User Manual for your application (*max. 2 A4 pages when printed*)
e) 2 sample levels/maps for the game

References:

http://en.wikipedia.org/wiki/Cannon_Fodder

Project Option H: A simple top-down perspective racing game

Your task is to write a simple top-down racing game (*using 3D graphics*) like the popular game MicroMachines (*description below* [*]) which must be playable (*but need not contain all of the original features*), using the OpenGL API, which must have the following features:

1. A custom file-format for tracks. Tracks can be interactively selected and loaded in by the user.
2. A user interface which is operated by the keyboard.
3. textured in-game 3D objects & environmental building blocks loaded from files of at least one of the following file formats:
   - DXF
   - Alias|Wavefront Object format
4. Either a 2-player mode or a simple AI opponent.

Extra features or other innovative content in terms of graphics (*design*) or additional in-game options (*power-ups etc.)* will be part of your overall mark.

You have to provide

a) One executable file & all necessary resource files (*textures etc.*)
b) All sourcecode files (*commented*) needed for compiling the project
c) Design & Implementation Documentation (*up to ca. 6 A4 pages*)
d) HTML User Manual for your application (*max. 2 A4 pages when printed*)
e) 2 sample tracks for the game

[*]MicroMachines

MicroMachines is one of the most innovative and fun racing games of all time. Based on the popular toys the aim of the game is to race small cars around a variety of circuits set inside the house. These include races around household appliances, on tables and worktops etc. The game only allows three controls - steer left, steer right and accellerate. Cars can bounce off obstacles and jump over ramps but all is drawn in a 2D top-down perspective.

References:

GameSpot review & screenshots: http://www.gamespot.com/pc/driving/micromachines2/

guidance notes for project preparation

If you are using Windows as your development platform that is acceptable, as long as you provide a Linux-compatible makefile for your project and your code compiles error-free under the Linux operating system.
When working on a programming project you may come across a problem which has already been solved by somebody other than you. In that case it is not necessary for you to "reinvent the wheel". However all work included in your project that was not produced by yourself must be properly credited in your report. If you use source code developed by somebody else, the sections of code that you use must be marked by comments within your project source code.

Example:

```
/* The following function was written by AUTHOR NAME and was found at
http://www.web.url */
void function()
{
...
}
```

Failure to properly credit your sources is a serious assessment offence and will be handled accordingly. This is not limited to the project source code but also extends to the documentation/report that goes with it:

Plagiarism: the copying or misappropriation of ideas (or their expression), text, software or data (or some combination thereof) without permission and due acknowledgement, i.e. the representation of another person's work as one's own or the use of another person's work without acknowledgement, e.g.

· the direct importation into one's work of more than a single phrase from another person's work without the use of quotation marks and identification of the source
· making a copy of all or part of another person's work and presenting it as one's own by failing to disclose the source
· making extensive use of another person's work, either by summarizing or paraphrasing it merely by changing a few words or altering the order of presentation, without acknowledgement
· the use of the ideas of another person without acknowledgement of the source, or the submission or presentation of work as one's own which is substantially the ideas or intellectual data of another

Other assessment offences relating to projects are:

Collusion: the representation of a piece of unauthorized group work as the work of a single student.

Commissioning another person to complete an assignment which is then submitted as the student's own work.

Computer fraud: the use of the material which belongs to another person and which is stored on a hard or floppy disk without acknowledgement and or without the written permission of the owner.

Duplication: the inclusion in coursework of any material which is identical or substantially similar to material which has already been submitted for any other assessment within the university or elsewhere (for example, the use of essay banks).

Academic Fraud: deliberate deception (*covers many of the above*).

The following URL provides information on how to reference other people's work:
http://www.bournemouth.ac.uk/library/using/guide_to_citing_internet_sourc.html

Report Template:

Report Title (*TimesNewRoman, Bold, 20pt*)

Your Name (*TimesNewRoman, 11pt*).

Abstract: blablabla this project is about this and that and I did this blabla (*about 50-100 word summary – single-spaced – TimesNewRoman, 12pt, justified*)

1. introduction (*TimesNewRoman 14pt*)

this is where you give a short introduction about the project and which aspects of the project you contributed to (*about 100-200 words – TimesNewRoman 11pt, 1½ spaced, justified)*

2. the project

this is where you describe in detail what you did for the project, how you did it and where you got your ideas from – references etc [1] … blablabla … This should not be confused with a "production diary", just listing what you did – lists like that should be avoided if possible or be put into an appendix. This section should contain elements of a design & implementation documentation.

Any references should be given using the same format [2]. References to websites should state the URL, followed by the date when the website was last accessed by you [3] (*TimesNewRoman 11pt, 1½ spaced, justified).*

3. conclusion

this is where you critically reflect on your work, what went right, what went wrong and what you have learned from this (*200-400 words*). The whole report (*excluding appendices*) should be about 2500 words in length (*this should translate to about 5-6 A4 pages*).

acknowledgements

a space for your academy awards acceptance speech… I would like to thank all my fellow students and my lecturers, supervisors, guardian angels, master Yoda … for supporting this project etc. etc.

references

[1] references that you refer to in the report (*books, programs, articles etc.*)

[2] Author(s) (year), chapter title (*if applicable*), book title, publisher, page numbers (*if known*)

[3] http://www.website.com - xx/xx/2007 (*date the link was last checked by you*)

appendices

additional documentation, file format descriptions etc. produced during the project