Advanced Programming in C++
Winter Term 2016/17 – Assignments # 2

Hasso Plattner Institute
Computer Graphics Systems Group

Prof. Dr. Jürgen Döllner

Objectives

In the assignments you will learn how to:

- apply C++ for procedural programming
- get aware of the specifics of floating point arithmetics
- use a profiler to optimize your code
- use streams for efficient data storage

For your programming tasks, it is not required to use common programming IDEs.

Task 2.1: Approximation of π

File: `montecarlopi.cpp`

Description: Implement a command-line program that computes an approximation of the value of π. For this purpose, you should use a variant of the "quarter of circle" algorithm, based on a Monte-Carlo approach:

Generate 2D points (x, y) in the unit square $[0, 1] \times [0, 1]$ (e.g., uniform distribution). Count how many of them are inside the corresponding quarter of the unit circle (distance to $(0, 0)$ is less than 1). The ratio between points inside the circle and the total number of points approximates the value of $\frac{\pi}{4}$. Because *many* iterations (the recomputation of π with one more sample point as before) are required, take care of arithmetic issues.

- How many iterations does your implementation need to get a medium error of $< 0.01\%$ for the last 10000 iterations (means the result is within $[3.1413, 3.1419]$)?
- List the most exact π representation using the C++ data-types `float`, `double`, and `long double` along with their storage in bytes for your system. How many bits of mantissa in a floating-point number are necessary to encode the first 20 decimal digits of π?
- You may use random-number generators or apply other sampling-based strategies to compute the ratio.
- For your implementation, use only elements of standard C++14 and, if needed, compiler extensions.
- Do not use numerics libraries or any other third-party libraries or classes that provide specialized floats.

Artifacts:

a) `montecarlopi.cpp` : Source code of the solution.

b) `montecarlopi.txt` : Results of the theoretical tasks (varying precisions for π).

Objectives: Procedural C++ elements, standard library functions, C++ floating-point arithmetics, random numbers.

Task 2.2: Business Card Raytracer

Files: `businesscard_raytracer.cpp`

Description: A famous C program, which fits on the back of a business card, implements a fully-functional basic ray tracer. This program, of course, is syntactically extremely compressed and dense; it is also based on the language features C provided at that time. Unscramble and rewrite this program using C++. Ensure that your implementation is functionally equivalent to the original one.

- Decipher and understand the code of the ray tracer.
- Rewrite the program using C++ and the Standard Library.
- Using the original scene, the refactored program should produce an (almost) identical result image.
- Use user-defined types, functions, and structures as well as the C++ standard library where appropriate.
- Take into account readability and maintenance.
- The size of your implementation, in common format, should not exceed 500 lines-of-code.
- Change the test scene to spell your name(s) and readjust the virtual camera so that the full scene is visible.

Artifacts:

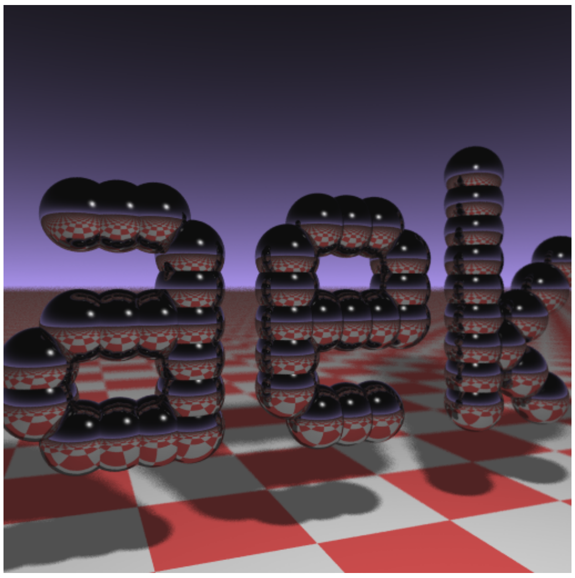
a) `businesscard_raytracer.cpp` : Source code of the business-card ray tracer.

b) `businesscard_raytracer.ppm` : A result image for your test scene being traced.

Example:

Listing 1: The source code of the business card raytracer that should be refactored.

```
#include <stdlib.h>   // card > aek.ppm
#include <stdio.h>
#include <math.h>
typedef int i;typedef float f;struct v{
f x,y,z;v operator+(v r){return v(x+r.x
,y+r.y,z+r.z);}v operator*(f r){return
v(x*r,y*r,z*r);}f operator%(v r){return
x*r.x+y*r.y+z*r.z;}v(){}v operator^(v r
){return v(y*r.z-z*r.y,z*r.x-x*r.z,x*r.
y-y*r.x);}v(f a,f b,f c){x=a;y=b;z=c;}v
operator!(){return*this*(1/sqrt(*this%*
this));}};i G[]={247570,280596,280600,
249748,18578,18577,231184,16,16};f R(){
return(f)rand()/RAND_MAX;}i T(v o,v d,f
&t,v&n){t=1e9;i m=0;f p=-o.z/d.z;if(.01
<p)t=p,n=v(0,0,1),m=1;for(i k=19;k--;)
for(i j=9;j--;)if(G[j]&1<<k){v p=o+v(-k
,0,-j-4);f b=p%d,c=p%p-1,q=b*b-c;if(q>0
){f s=-b-sqrt(q);if(s<t&&s>.01)t=s,n=!(
p+d*t),m=2;}}return m;}v S(v o,v d){f t
;v n;i m=T(o,d,t,n);if(!m)return v(.7,
.6,1)*pow(1-d.z,4);v h=o+d*t,l=!(v(9+R(
),9+R(),16)+h*-1),r=d+n*(n%d*-2);f b=l%
n;if(b<0||T(h,l,t,n))b=0;f p=pow(l%r*(b
>0),99);if(m&1){h=h*.2;return((i)(ceil(
h.x)+ceil(h.y))&1?v(3,1,1):v(3,3,3))*(b
*.2+.1);}return v(p,p,p)+S(h,r)*.5;}i
main(){printf("P6 512 512 255 ");v g=!v
(-6,-16,0),a=!(v(0,0,1)^g)*.002,b=!(g^a
)*.002,c=(a+b)*-256+g;for(i y=512;y--;)
for(i x=512;x--;){v p(13,13,13);for(i r
=64;r--;){v t=a*(R()-.5)*99+b*(R()-.5)*
99;p=S(v(17,16,8)+t,!(t*-1+(a*(R()+x)+b
*(y+R())+c)*16))*3.5+p;}printf("%c%c%c"
,(i)p.x,(i)p.y,(i)p.z);}}
```


Raytraced example scene of the business card raytracer.

Objectives: Program comprehension, Reverse engineering, refactoring, C programming style, C++ procedural programming.

Task 2.3: File Formats with Key-Value Pair Information and Raw Data

Files: `textfileformat.cpp`, `textfileformat.h`, `binaryfileformat.cpp`, `binaryfileformat.h`, `FileContent.cpp`, `FileContent.h`, `Value.cpp`, `Value.h`, `fileformat-test.cpp`, `fileformat-testdata.cpp`, `fileformat-testdata.h`

Description: Implement a statically linkable library that handles the encoding and decoding of two specific file formats (text-based and binary). Both file formats contain a section for key-value based information and a section with application-specific raw data that is not specified by the file format. The key-value section of the file format may be empty and contains data as follows:

key The key is a character sequence with only lower-case letters and optional dashes between characters (neither at the beginning nor on the end are permitted). Thereby, it can be represented as variable-length string. The maximum number of characters for a key is 2^{30}.

value The value can be of type

- boolean,
- 32-bit signed integer number,
- 32-bit floating-point number,
- variable-length string (maximum length is 2^{30}), permitted characters:
  - digits: `0123456789`
  - letters: `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ`
  - punctuation: `!#$%&'()*+,-./:;<=>?@[]^_`‘`{}~|`
  - whitespace: ␣

The two specific file formats are encoded as follows:

Text-based It is a line-based format (delimiter is the Unix line-break LF) where each line contains the key-value pair, separated by the equal sign "=". One empty line marks the beginning of the raw data section. The value type is inferred by the following rules:

boolean The value must match either `true` or `false`

integer number The value is encoded using an optional sign character (plus "+" or minus "-") and subsequent digits. Only decimal encoded values are permitted.

floating-point number The values are encoded either as decimal numbers with a decimal period or in scientific notation.

string Strings are encoded with enclosing quotes (example: "test string"). Escaping of quotes inside the string is not required.

Binary The format specifies the 16-bit magic number 0x95CC for the beginning of the file to identify the binary file format. The upcoming 8 bytes are used to specify the index of the first byte of the raw data section. The first section is the key-value pair section containing types data with keys as identifiers. The second section is the raw data section and can be immediately read by seeking to the position specified by the bytes 3 to 10. The key-value pairs are encoded without explicit delimiters. Therefore, a pair is encoded using the first 2 bits as the type of the value (0 for boolean, 1 for signed integer number, 2 for floating-point number and 3 for variable-length string), the upcoming 30 bits for the number of characters of the remaining string for the key. After that, the variable number of characters of the key are encoded using 8-bit characters. The value is encoded using the following formats:

boolean Encoded using 1 byte where 0 bits set are interpreted as false and all other values are interpreted as true. Serializers are advised to encode a true as least-significant bit set and all other bits unset.

signed integer number Encoded using signed 32-bit two's-complement.

floating-point number Encoded using IEEE 754 32-bit floating-point format.

string Encoded with the number of characters of the string in the first 4 bytes (maximum number is 2^{30}) and following 8-bit characters.

The variable-length strings have no trailing zero-byte.

The provided test can be used to validate the implementations and to compare the run-time performances of the approaches.

Tasks:

a) Implement deserialization of the text-based file format.

b) Implement deserialization of the binary file format.

c) Implement serialization of the text-based file format.

d) Implement serialization of the binary file format.

e) Use C++ standard library facilities (e.g., I/O streams) to read, parse and write the data.

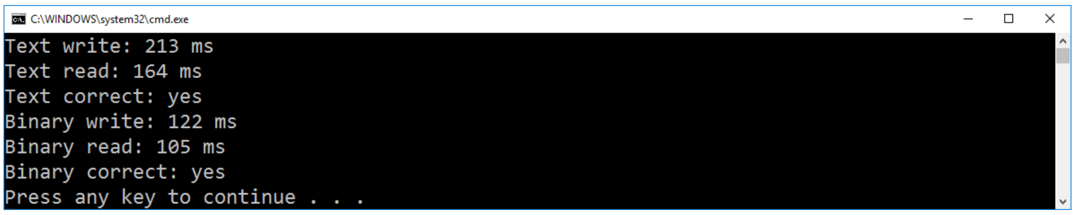
f) Measure the run-time performance of the four implementations using the provided test.

Artifacts:

a) `textfileformat.cpp`, `binaryfileformat.cpp` : Source code of the solution.

b) `fileformat-test.txt` : Command-line output of the provided test for a dataset with 100,000 key-value pairs and 20 MiB of raw data.

Example:



```
Text write: 213 ms
Text read: 164 ms
Text correct: yes
Binary write: 122 ms
Binary read: 105 ms
Binary correct: yes
Press any key to continue . . .
```

Objectives: Forward engineering, Binary encoding of data, C++ data type representations, I/O streams, Parser logic, Serializer logic.

Instructions

Pair Programming On these assignments, you are encouraged (not required) to work with a partner provided you practice pair programming. Pair programming "is a practice in which two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code, or test." One partner is driving (designing and typing the code) while the other is navigating (reviewing the work, identifying bugs, and asking questions). The two partners switch roles every 30-40 minutes, and on demand, brainstorm.

Cross-Platform The assignments can be solved on all major platforms (i.e., Windows, Linux, macOS). The evaluation of submitted assignments can be carried out on any of these platforms. All results are required to be cross-platform, that is, they compile successfully and run indifferently with respect to their input and output.

Upload Results All described artifacts are submitted to the course moodle system `https://moodle.hpi3d.de/mod/assignment/view.php?id=2678` as a zipped archive with the following naming convention: `assignment_2_matrikNr1.zip` or `assignment_2_matrikNr1_matrikNr2.zip` whether or not pair programming was applied. Compiled, intermediate, or temporary files should not be included (`*.obj, *.pdb, *.ilk, *.ncb` etc.). If pair programming is used, the results are turned in only once. The assignments #2 are due to the next tutorial on November 16th, 9:15 a.m.

Violation of Rules a violation of rules results in grading the affected assignments with 0 points.

- Writing code with a partner without following the pair programming instructions listed above (e.g., if one partner does not participate in the process) is a serious violation of the course collaboration policy.
- Plagiarism represents a serious violation of the course policy.