ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ "ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

Факультет ФИСП

Кафедра ПИ

Лабораторная работа №7

предмет: «Технология разработки и сопровождения интернет-сайтов»

по теме: «Работа с файловой системой»

Вариант №15

Выполнил: ст. гр. ПИ-21б

Духнич Д.И.

Проверил:

Коломойцева И.А.

ДОНЕЦК – 2024

1. Цель работы

Изучить возможности языка PHP в вопросе создания PDF-файлов

1. Задание

Пройти по дереву каталогов, начиная с текущего, и удалить все файлы на языке Си, содержащие внутри максимальное количество операторов if-else не менее 2, и имеющие шаблон имени файла: в названии файла имеются две точки и дата в виде YYMMDD в любом месте и младше месяца и старше недели с правами только на чтение

1. Порядок выполнения
2. Написать PHP-скрипт в соответствии с заданным вариантом.
3. Протестировать созданный PHP-скрипт.
4. Демонстрация работы скрипта:

На рисунках 1.1 – 1.2 изображена демонстрация работы скрипта.



Рисунок 1.1 – Демонстрация работы скрипта

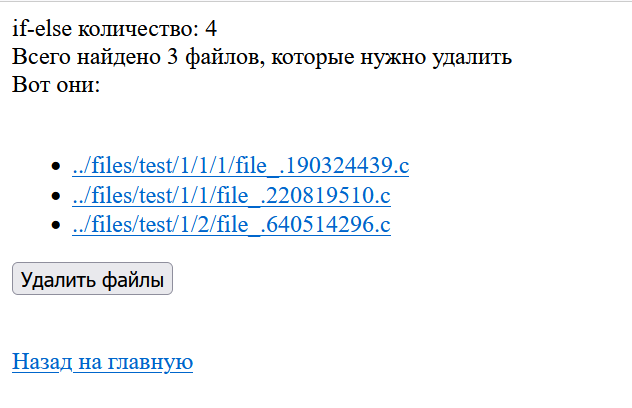


Рисунок 1.2 – Демонстрация работы скрипта

5 Код программы

.\autoload.php:

<?php

define('\_\_ROOT\_\_', dirname(\_\_FILE\_\_, 2));

function getManager($classname) {

$classname = str\_replace('\\', '/', $classname);

require\_once(\_\_ROOT\_\_. '/'. $classname . '.php');

}

spl\_autoload\_register('getManager');

.\CatalogManager.php:

<?php

namespace lab7;

// 15. Пройти по дереву каталогов, начиная с текущего, и удалить все

// файлы на языке Си, содержащие внутри максимальное количество

// операторов if-else не менее 2, и имеющие шаблон имени файла: в названии

// файла имеются две точки и дата в виде YYMMDD в любом месте и младше

// месяца и старше недели с правами только на чтение.

use DateTime;

class CatalogManager

{

private $files\_needs\_to\_be\_deleted = [];

private $temp\_files = [];

public function index()

{

header("Location: ./views/index.php");

die();

}

private function deleteFiles($dir)

{

$files = glob($dir.'/\*');

foreach ($files as $file) {

if (is\_file($file)) {

unlink($file);

} else {

$this->deleteFiles($file);

rmdir($file);

}

}

}

public function generate()

{

$directory = dirname(\_\_FILE\_\_, 2).'/lab7/files/';

$start\_date = strtotime('2023-02-01');

$end\_date = strtotime(date('Y-m-d'));

$dirs = [

'./1', './2', './3', './test', './test/1', './test/2', './test/3', './test/1/1', './test/1/2', './test/1/1/1',

'./test/1/1/2', './1/5', './abc', './abc/qwe', './abc/ewq', './test/test'

];

$this->deleteFiles($directory);

foreach ($dirs as $dir) {

mkdir($directory.$dir);

}

for ($i = 1; $i <= 10000; $i++) {

$target\_file = mt\_rand(1, 6);

$random\_dir = $dirs[rand(0, count($dirs) - 1)];

$filename = $random\_dir.'/file\_'.(mt\_rand(0, 1) < 0.5 ? '.' : '\_').mt\_rand(100,

999999999).'.c';

$filepath = $directory.$filename;

if (mt\_rand(0, 1) < 0.5) {

$file\_rights = 0444; // read-only

} else {

$file\_rights = 0777; // all rights

}

$file\_creation\_time = mt\_rand($start\_date, $end\_date);

$content = file\_get\_contents('../examples/file'.$target\_file.'.c');

file\_put\_contents($filepath, $content);

chmod($filepath, $file\_rights);

exec("touch -t ".date('YmdHi', $file\_creation\_time)." '$filepath'");

}

header("Location: ../views/index.php");

die();

}

public function listFolders($dir): string

{

$html = "<ul>";

$files = scandir($dir);

foreach ($files as $file) {

if ($file != "." && $file != "..") {

if (is\_dir($dir."/".$file)) {

$html .= "<li>".$file;

$html .= $this->listFolders($dir."/".$file);

$html .= "</li>";

} else {

$file\_path = $dir."/".$file;

$file\_info = stat($file\_path);

$creation\_date = date("Y-m-d H:i:s", $file\_info['mtime']);

$user\_abilities = decoct(fileperms($file\_path) & 0777);

$html .= '<li><a href="'.$file\_path.'">'.$file.' - '.$creation\_date.', '.$user\_abilities.'</a></li>';

}

}

}

$html .= "</ul>";

return $html;

}

private function is\_filename\_valid($filename): bool

{

if (substr\_count($filename, '.') != 2) {

return false;

}

preg\_match('/\d{6}/', $filename, $matches);

if (!$matches) {

return false;

}

foreach ($matches as $date\_string) {

$date\_string = strval($date\_string);

$date = DateTime::createFromFormat('ymd', $date\_string);

if ($date && $date->format('ymd') === $date\_string) {

return true;

}

}

return false;

}

private function deleteRec($dir)

{

$files = scandir($dir);

foreach ($files as $file) {

if ($file != "." && $file != "..") {

$file\_path = $dir."/".$file;

if (is\_dir($file\_path)) {

$this->deleteRec($file\_path);

} else {

$file\_info = stat($file\_path);

$user\_abilities = decoct(fileperms($file\_path) & 0777);

if ($user\_abilities == '444') {

$from\_creation = (time() - filemtime($file\_path));

if ($from\_creation > 24 \* 60 \* 60 \* 7 && $from\_creation < 24 \* 60 \* 60 \* 30) {

if ($this->is\_filename\_valid($file)) {

$content = file\_get\_contents($file\_path);

$count = substr\_count($content, 'if') + substr\_count($content, 'else');

$this->temp\_files[] = [

'count' => $count,

'file\_path' => $file\_path

];

}

}

}

}

}

}

}

public function delete()

{

$this->deleteRec('../files');

$max\_count = 0;

foreach ($this->temp\_files as $file\_data) {

$max\_count = max($max\_count, $file\_data['count']);

}

echo 'if-else количество: '.$max\_count.'<br>';

foreach ($this->temp\_files as $file\_data) {

if ($max\_count == $file\_data['count'] && $max\_count >= 2) {

$this->files\_needs\_to\_be\_deleted[] = $file\_data['file\_path'];

}

}

if (count($this->files\_needs\_to\_be\_deleted)) {

echo "Всего найдено ".count($this->files\_needs\_to\_be\_deleted).' файлов, которые нужно удалить <br>';

echo 'Вот они: <br><br><ul>';

foreach ($this->files\_needs\_to\_be\_deleted as $file) {

echo '<li><a href="'.$file.'">'.$file.'</a></li>';

}

echo '</ul>';

echo '<form action="./remove.php" method="post">

<input type="hidden" name="remove" value="'.base64\_encode(json\_encode($this->files\_needs\_to\_be\_deleted)).'">

<button type="submit">Удалить файлы</button>

</form><br>';

} else {

echo 'Нечего удалять. Уходите отсюда. <br>';

}

echo '<a href="../index.php">Назад на главную</a>';

}

public function remove($files)

{

$files\_needs\_to\_be\_deleted = json\_decode(base64\_decode($files));

foreach ($files\_needs\_to\_be\_deleted as $file\_path) {

if (file\_exists($file\_path)) {

unlink($file\_path);

}

}

header("Location: ../index.php");

die();

}

}

.\data.txt:

.\index.php:

<?php

require\_once('autoload.php');

use lab7\CatalogManager;

$catalogManager = new CatalogManager();

$catalogManager->index();

.\Untitled.ipynb:

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"id": "2c828750",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"autoload.php\n",

"CatalogManager.php\n",

"index.php\n",

"Untitled.ipynb\n"

]

}

],

"source": [

"import os\n",

"files = [f for f in os.listdir('.') if os.path.isfile(f)]\n",

"for f in files:\n",

" print(f)\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "609ffd14",

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.11.3"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}

.\.ipynb\_checkpoints\Untitled-checkpoint.ipynb:

{

"cells": [],

"metadata": {},

"nbformat": 4,

"nbformat\_minor": 5

}

.\controllers\delete.php:

<?php

require\_once('../autoload.php');

use lab7\CatalogManager;

$catalogManager = new CatalogManager();

$catalogManager->delete();

.\controllers\generate.php:

<?php

require\_once('../autoload.php');

use lab7\CatalogManager;

$catalogManager = new CatalogManager();

$catalogManager->generate();

.\controllers\remove.php:

<?php

require\_once('../autoload.php');

use lab7\CatalogManager;

$catalogManager = new CatalogManager();

$catalogManager->remove($\_POST['remove']);

.\examples\file1.c:

#!../c -lglut -lGLU -lGL -lXmu -lXext -lX11 -lm --

/\* Copyright (c) Mark J. Kilgard, 1997. \*/

/\* This program is freely distributable without licensing fees

and is provided without guarantee or warrantee expressed or

implied. This program is -not- in the public domain. \*/

/\* This program was requested by Patrick Earl; hopefully someone else

will write the equivalent Direct3D immediate mode program. \*/

#include <GL/glut.h>

GLfloat light\_diffuse[] = {1.0, 0.0, 0.0, 1.0}; /\* Red diffuse light. \*/

GLfloat light\_position[] = {1.0, 1.0, 1.0, 0.0}; /\* Infinite light location. \*/

GLfloat n[6][3] = { /\* Normals for the 6 faces of a cube. \*/

{-1.0, 0.0, 0.0}, {0.0, 1.0, 0.0}, {1.0, 0.0, 0.0},

{0.0, -1.0, 0.0}, {0.0, 0.0, 1.0}, {0.0, 0.0, -1.0} };

GLint faces[6][4] = { /\* Vertex indices for the 6 faces of a cube. \*/

{0, 1, 2, 3}, {3, 2, 6, 7}, {7, 6, 5, 4},

{4, 5, 1, 0}, {5, 6, 2, 1}, {7, 4, 0, 3} };

GLfloat v[8][3]; /\* Will be filled in with X,Y,Z vertexes. \*/

void

drawBox(void)

{

int i;

for (i = 0; i < 6; i++) {

glBegin(GL\_QUADS);

glNormal3fv(&n[i][0]);

glVertex3fv(&v[faces[i][0]][0]);

glVertex3fv(&v[faces[i][1]][0]);

glVertex3fv(&v[faces[i][2]][0]);

glVertex3fv(&v[faces[i][3]][0]);

glEnd();

}

}

void

display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

drawBox();

glutSwapBuffers();

}

void

init(void)

{

/\* Setup cube vertex data. \*/

v[0][0] = v[1][0] = v[2][0] = v[3][0] = -1;

v[4][0] = v[5][0] = v[6][0] = v[7][0] = 1;

v[0][1] = v[1][1] = v[4][1] = v[5][1] = -1;

v[2][1] = v[3][1] = v[6][1] = v[7][1] = 1;

v[0][2] = v[3][2] = v[4][2] = v[7][2] = 1;

v[1][2] = v[2][2] = v[5][2] = v[6][2] = -1;

/\* Enable a single OpenGL light. \*/

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

/\* Use depth buffering for hidden surface elimination. \*/

glEnable(GL\_DEPTH\_TEST);

/\* Setup the view of the cube. \*/

glMatrixMode(GL\_PROJECTION);

gluPerspective( /\* field of view in degree \*/ 40.0,

/\* aspect ratio \*/ 1.0,

/\* Z near \*/ 1.0, /\* Z far \*/ 10.0);

glMatrixMode(GL\_MODELVIEW);

gluLookAt(0.0, 0.0, 5.0, /\* eye is at (0,0,5) \*/

0.0, 0.0, 0.0, /\* center is at (0,0,0) \*/

0.0, 1.0, 0.); /\* up is in positive Y direction \*/

/\* Adjust cube position to be asthetic angle. \*/

glTranslatef(0.0, 0.0, -1.0);

glRotatef(60, 1.0, 0.0, 0.0);

glRotatef(-20, 0.0, 0.0, 1.0);

}

int

main(int argc, char \*\*argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutCreateWindow("red 3D lighted cube");

glutDisplayFunc(display);

init();

glutMainLoop();

return 0; /\* ANSI C requires main to return int. \*/

}

.\examples\file2.c:

#!../c -lglut -lGLU -lGL -lXmu -lXext -lX11 -lm --

/\* Copyright (c) Mark J. Kilgard, 1994, 1997. \*/

/\* This program is freely distributable without licensing fees

and is provided without guarantee or warrantee expressed or

implied. This program is -not- in the public domain. \*/

/\* Example for PC game developers to show how to \*combine\* texturing,

reflections, and projected shadows all in real-time with OpenGL.

Robust reflections use stenciling. Robust projected shadows

use both stenciling and polygon offset. PC game programmers

should realize that neither stenciling nor polygon offset are

supported by Direct3D, so these real-time rendering algorithms

are only really viable with OpenGL.

The program has modes for disabling the stenciling and polygon

offset uses. It is worth running this example with these features

toggled off so you can see the sort of artifacts that result.

Notice that the floor texturing, reflections, and shadowing

all co-exist properly. \*/

/\* When you run this program: Left mouse button controls the

view. Middle mouse button controls light position (left &

right rotates light around dino; up & down moves light

position up and down). Right mouse button pops up menu. \*/

/\* Check out the comments in the "redraw" routine to see how the

reflection blending and surface stenciling is done. You can

also see in "redraw" how the projected shadows are rendered,

including the use of stenciling and polygon offset. \*/

/\* This program is derived from glutdino.c \*/

/\* Compile: cc -o dinoshade dinoshade.c -lglut -lGLU -lGL -lXmu -lXext -lX11 -lm \*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h> /\* for cos(), sin(), and sqrt() \*/

#include <GL/glut.h> /\* OpenGL Utility Toolkit header \*/

/\* Some <math.h> files do not define M\_PI... \*/

#ifndef M\_PI

#define M\_PI 3.14159265

#endif

/\* Variable controlling various rendering modes. \*/

static int stencilReflection = 1, stencilShadow = 1, offsetShadow = 1;

static int renderShadow = 1, renderDinosaur = 1, renderReflection = 1;

static int linearFiltering = 0, useMipmaps = 0, useTexture = 1;

static int reportSpeed = 0;

static int animation = 1;

static GLboolean lightSwitch = GL\_TRUE;

static int directionalLight = 1;

static int forceExtension = 0;

/\* Time varying or user-controled variables. \*/

static float jump = 0.0;

static float lightAngle = 0.0, lightHeight = 20;

GLfloat angle = -150; /\* in degrees \*/

GLfloat angle2 = 30; /\* in degrees \*/

int moving, startx, starty;

int lightMoving = 0, lightStartX, lightStartY;

enum {

MISSING, EXTENSION, ONE\_DOT\_ONE

};

int polygonOffsetVersion;

static GLdouble bodyWidth = 3.0;

/\* \*INDENT-OFF\* \*/

static GLfloat body[][2] = { {0, 3}, {1, 1}, {5, 1}, {8, 4}, {10, 4}, {11, 5},

{11, 11.5}, {13, 12}, {13, 13}, {10, 13.5}, {13, 14}, {13, 15}, {11, 16},

{8, 16}, {7, 15}, {7, 13}, {8, 12}, {7, 11}, {6, 6}, {4, 3}, {3, 2},

{1, 2} };

static GLfloat arm[][2] = { {8, 10}, {9, 9}, {10, 9}, {13, 8}, {14, 9}, {16, 9},

{15, 9.5}, {16, 10}, {15, 10}, {15.5, 11}, {14.5, 10}, {14, 11}, {14, 10},

{13, 9}, {11, 11}, {9, 11} };

static GLfloat leg[][2] = { {8, 6}, {8, 4}, {9, 3}, {9, 2}, {8, 1}, {8, 0.5}, {9, 0},

{12, 0}, {10, 1}, {10, 2}, {12, 4}, {11, 6}, {10, 7}, {9, 7} };

static GLfloat eye[][2] = { {8.75, 15}, {9, 14.7}, {9.6, 14.7}, {10.1, 15},

{9.6, 15.25}, {9, 15.25} };

static GLfloat lightPosition[4];

static GLfloat lightColor[] = {0.8, 1.0, 0.8, 1.0}; /\* green-tinted \*/

static GLfloat skinColor[] = {0.1, 1.0, 0.1, 1.0}, eyeColor[] = {1.0, 0.2, 0.2, 1.0};

/\* \*INDENT-ON\* \*/

/\* Nice floor texture tiling pattern. \*/

static char \*circles[] = {

"....xxxx........",

"..xxxxxxxx......",

".xxxxxxxxxx.....",

".xxx....xxx.....",

"xxx......xxx....",

"xxx......xxx....",

"xxx......xxx....",

"xxx......xxx....",

".xxx....xxx.....",

".xxxxxxxxxx.....",

"..xxxxxxxx......",

"....xxxx........",

"................",

"................",

"................",

"................",

};

static void

makeFloorTexture(void)

{

GLubyte floorTexture[16][16][3];

GLubyte \*loc;

int s, t;

/\* Setup RGB image for the texture. \*/

loc = (GLubyte\*) floorTexture;

for (t = 0; t < 16; t++) {

for (s = 0; s < 16; s++) {

if (circles[t][s] == 'x') {

/\* Nice green. \*/

loc[0] = 0x1f;

loc[1] = 0x8f;

loc[2] = 0x1f;

} else {

/\* Light gray. \*/

loc[0] = 0xaa;

loc[1] = 0xaa;

loc[2] = 0xaa;

}

loc += 3;

}

}

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

if (useMipmaps) {

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER,

GL\_LINEAR\_MIPMAP\_LINEAR);

gluBuild2DMipmaps(GL\_TEXTURE\_2D, 3, 16, 16,

GL\_RGB, GL\_UNSIGNED\_BYTE, floorTexture);

} else {

if (linearFiltering) {

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

} else {

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

}

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, 16, 16, 0,

GL\_RGB, GL\_UNSIGNED\_BYTE, floorTexture);

}

}

enum {

X, Y, Z, W

};

enum {

A, B, C, D

};

/\* Create a matrix that will project the desired shadow. \*/

void

shadowMatrix(GLfloat shadowMat[4][4],

GLfloat groundplane[4],

GLfloat lightpos[4])

{

GLfloat dot;

/\* Find dot product between light position vector and ground plane normal. \*/

dot = groundplane[X] \* lightpos[X] +

groundplane[Y] \* lightpos[Y] +

groundplane[Z] \* lightpos[Z] +

groundplane[W] \* lightpos[W];

shadowMat[0][0] = dot - lightpos[X] \* groundplane[X];

shadowMat[1][0] = 0.f - lightpos[X] \* groundplane[Y];

shadowMat[2][0] = 0.f - lightpos[X] \* groundplane[Z];

shadowMat[3][0] = 0.f - lightpos[X] \* groundplane[W];

shadowMat[X][1] = 0.f - lightpos[Y] \* groundplane[X];

shadowMat[1][1] = dot - lightpos[Y] \* groundplane[Y];

shadowMat[2][1] = 0.f - lightpos[Y] \* groundplane[Z];

shadowMat[3][1] = 0.f - lightpos[Y] \* groundplane[W];

shadowMat[X][2] = 0.f - lightpos[Z] \* groundplane[X];

shadowMat[1][2] = 0.f - lightpos[Z] \* groundplane[Y];

shadowMat[2][2] = dot - lightpos[Z] \* groundplane[Z];

shadowMat[3][2] = 0.f - lightpos[Z] \* groundplane[W];

shadowMat[X][3] = 0.f - lightpos[W] \* groundplane[X];

shadowMat[1][3] = 0.f - lightpos[W] \* groundplane[Y];

shadowMat[2][3] = 0.f - lightpos[W] \* groundplane[Z];

shadowMat[3][3] = dot - lightpos[W] \* groundplane[W];

}

/\* Find the plane equation given 3 points. \*/

void

findPlane(GLfloat plane[4],

GLfloat v0[3], GLfloat v1[3], GLfloat v2[3])

{

GLfloat vec0[3], vec1[3];

/\* Need 2 vectors to find cross product. \*/

vec0[X] = v1[X] - v0[X];

vec0[Y] = v1[Y] - v0[Y];

vec0[Z] = v1[Z] - v0[Z];

vec1[X] = v2[X] - v0[X];

vec1[Y] = v2[Y] - v0[Y];

vec1[Z] = v2[Z] - v0[Z];

/\* find cross product to get A, B, and C of plane equation \*/

plane[A] = vec0[Y] \* vec1[Z] - vec0[Z] \* vec1[Y];

plane[B] = -(vec0[X] \* vec1[Z] - vec0[Z] \* vec1[X]);

plane[C] = vec0[X] \* vec1[Y] - vec0[Y] \* vec1[X];

plane[D] = -(plane[A] \* v0[X] + plane[B] \* v0[Y] + plane[C] \* v0[Z]);

}

void

extrudeSolidFromPolygon(GLfloat data[][2], unsigned int dataSize,

GLdouble thickness, GLuint side, GLuint edge, GLuint whole)

{

static GLUtriangulatorObj \*tobj = NULL;

GLdouble vertex[3], dx, dy, len;

int i;

int count = (int) (dataSize / (2 \* sizeof(GLfloat)));

if (tobj == NULL) {

tobj = gluNewTess(); /\* create and initialize a GLU

polygon tessellation object \*/

gluTessCallback(tobj, GLU\_BEGIN, glBegin);

gluTessCallback(tobj, GLU\_VERTEX, glVertex2fv); /\* semi-tricky \*/

gluTessCallback(tobj, GLU\_END, glEnd);

}

glNewList(side, GL\_COMPILE);

glShadeModel(GL\_SMOOTH); /\* smooth minimizes seeing

tessellation \*/

gluBeginPolygon(tobj);

for (i = 0; i < count; i++) {

vertex[0] = data[i][0];

vertex[1] = data[i][1];

vertex[2] = 0;

gluTessVertex(tobj, vertex, data[i]);

}

gluEndPolygon(tobj);

glEndList();

glNewList(edge, GL\_COMPILE);

glShadeModel(GL\_FLAT); /\* flat shade keeps angular hands

from being "smoothed" \*/

glBegin(GL\_QUAD\_STRIP);

for (i = 0; i <= count; i++) {

/\* mod function handles closing the edge \*/

glVertex3f(data[i % count][0], data[i % count][1], 0.0);

glVertex3f(data[i % count][0], data[i % count][1], thickness);

/\* Calculate a unit normal by dividing by Euclidean

distance. We \* could be lazy and use

glEnable(GL\_NORMALIZE) so we could pass in \* arbitrary

normals for a very slight performance hit. \*/

dx = data[(i + 1) % count][1] - data[i % count][1];

dy = data[i % count][0] - data[(i + 1) % count][0];

len = sqrt(dx \* dx + dy \* dy);

glNormal3f(dx / len, dy / len, 0.0);

}

glEnd();

glEndList();

glNewList(whole, GL\_COMPILE);

glFrontFace(GL\_CW);

glCallList(edge);

glNormal3f(0.0, 0.0, -1.0); /\* constant normal for side \*/

glCallList(side);

glPushMatrix();

glTranslatef(0.0, 0.0, thickness);

glFrontFace(GL\_CCW);

glNormal3f(0.0, 0.0, 1.0); /\* opposite normal for other side \*/

glCallList(side);

glPopMatrix();

glEndList();

}

/\* Enumerants for refering to display lists. \*/

typedef enum {

RESERVED, BODY\_SIDE, BODY\_EDGE, BODY\_WHOLE, ARM\_SIDE, ARM\_EDGE, ARM\_WHOLE,

LEG\_SIDE, LEG\_EDGE, LEG\_WHOLE, EYE\_SIDE, EYE\_EDGE, EYE\_WHOLE

} displayLists;

static void

makeDinosaur(void)

{

extrudeSolidFromPolygon(body, sizeof(body), bodyWidth,

BODY\_SIDE, BODY\_EDGE, BODY\_WHOLE);

extrudeSolidFromPolygon(arm, sizeof(arm), bodyWidth / 4,

ARM\_SIDE, ARM\_EDGE, ARM\_WHOLE);

extrudeSolidFromPolygon(leg, sizeof(leg), bodyWidth / 2,

LEG\_SIDE, LEG\_EDGE, LEG\_WHOLE);

extrudeSolidFromPolygon(eye, sizeof(eye), bodyWidth + 0.2,

EYE\_SIDE, EYE\_EDGE, EYE\_WHOLE);

}

static void

drawDinosaur(void)

{

glPushMatrix();

/\* Translate the dinosaur to be at (0,8,0). \*/

glTranslatef(-8, 0, -bodyWidth / 2);

glTranslatef(0.0, jump, 0.0);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, skinColor);

glCallList(BODY\_WHOLE);

glTranslatef(0.0, 0.0, bodyWidth);

glCallList(ARM\_WHOLE);

glCallList(LEG\_WHOLE);

glTranslatef(0.0, 0.0, -bodyWidth - bodyWidth / 4);

glCallList(ARM\_WHOLE);

glTranslatef(0.0, 0.0, -bodyWidth / 4);

glCallList(LEG\_WHOLE);

glTranslatef(0.0, 0.0, bodyWidth / 2 - 0.1);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, eyeColor);

glCallList(EYE\_WHOLE);

glPopMatrix();

}

static GLfloat floorVertices[4][3] = {

{ -20.0, 0.0, 20.0 },

{ 20.0, 0.0, 20.0 },

{ 20.0, 0.0, -20.0 },

{ -20.0, 0.0, -20.0 },

};

/\* Draw a floor (possibly textured). \*/

static void

drawFloor(void)

{

glDisable(GL\_LIGHTING);

if (useTexture) {

glEnable(GL\_TEXTURE\_2D);

}

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0);

glVertex3fv(floorVertices[0]);

glTexCoord2f(0.0, 16.0);

glVertex3fv(floorVertices[1]);

glTexCoord2f(16.0, 16.0);

glVertex3fv(floorVertices[2]);

glTexCoord2f(16.0, 0.0);

glVertex3fv(floorVertices[3]);

glEnd();

if (useTexture) {

glDisable(GL\_TEXTURE\_2D);

}

glEnable(GL\_LIGHTING);

}

static GLfloat floorPlane[4];

static GLfloat floorShadow[4][4];

static void

redraw(void)

{

int start, end;

if (reportSpeed) {

start = glutGet(GLUT\_ELAPSED\_TIME);

}

/\* Clear; default stencil clears to zero. \*/

if ((stencilReflection && renderReflection) || (stencilShadow && renderShadow)) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT | GL\_STENCIL\_BUFFER\_BIT);

} else {

/\* Avoid clearing stencil when not using it. \*/

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

}

/\* Reposition the light source. \*/

lightPosition[0] = 12\*cos(lightAngle);

lightPosition[1] = lightHeight;

lightPosition[2] = 12\*sin(lightAngle);

if (directionalLight) {

lightPosition[3] = 0.0;

} else {

lightPosition[3] = 1.0;

}

shadowMatrix(floorShadow, floorPlane, lightPosition);

glPushMatrix();

/\* Perform scene rotations based on user mouse input. \*/

glRotatef(angle2, 1.0, 0.0, 0.0);

glRotatef(angle, 0.0, 1.0, 0.0);

/\* Tell GL new light source position. \*/

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPosition);

if (renderReflection) {

if (stencilReflection) {

/\* We can eliminate the visual "artifact" of seeing the "flipped"

dinosaur underneath the floor by using stencil. The idea is

draw the floor without color or depth update but so that

a stencil value of one is where the floor will be. Later when

rendering the dinosaur reflection, we will only update pixels

with a stencil value of 1 to make sure the reflection only

lives on the floor, not below the floor. \*/

/\* Don't update color or depth. \*/

glDisable(GL\_DEPTH\_TEST);

glColorMask(GL\_FALSE, GL\_FALSE, GL\_FALSE, GL\_FALSE);

/\* Draw 1 into the stencil buffer. \*/

glEnable(GL\_STENCIL\_TEST);

glStencilOp(GL\_REPLACE, GL\_REPLACE, GL\_REPLACE);

glStencilFunc(GL\_ALWAYS, 1, 0xffffffff);

/\* Now render floor; floor pixels just get their stencil set to 1. \*/

drawFloor();

/\* Re-enable update of color and depth. \*/

glColorMask(GL\_TRUE, GL\_TRUE, GL\_TRUE, GL\_TRUE);

glEnable(GL\_DEPTH\_TEST);

/\* Now, only render where stencil is set to 1. \*/

glStencilFunc(GL\_EQUAL, 1, 0xffffffff); /\* draw if ==1 \*/

glStencilOp(GL\_KEEP, GL\_KEEP, GL\_KEEP);

}

glPushMatrix();

/\* The critical reflection step: Reflect dinosaur through the floor

(the Y=0 plane) to make a relection. \*/

glScalef(1.0, -1.0, 1.0);

/\* Reflect the light position. \*/

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPosition);

/\* To avoid our normals getting reversed and hence botched lighting

on the reflection, turn on normalize. \*/

glEnable(GL\_NORMALIZE);

glCullFace(GL\_FRONT);

/\* Draw the reflected dinosaur. \*/

drawDinosaur();

/\* Disable noramlize again and re-enable back face culling. \*/

glDisable(GL\_NORMALIZE);

glCullFace(GL\_BACK);

glPopMatrix();

/\* Switch back to the unreflected light position. \*/

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPosition);

if (stencilReflection) {

glDisable(GL\_STENCIL\_TEST);

}

}

/\* Back face culling will get used to only draw either the top or the

bottom floor. This let's us get a floor with two distinct

appearances. The top floor surface is reflective and kind of red.

The bottom floor surface is not reflective and blue. \*/

/\* Draw "bottom" of floor in blue. \*/

glFrontFace(GL\_CW); /\* Switch face orientation. \*/

glColor4f(0.1, 0.1, 0.7, 1.0);

drawFloor();

glFrontFace(GL\_CCW);

if (renderShadow) {

if (stencilShadow) {

/\* Draw the floor with stencil value 3. This helps us only

draw the shadow once per floor pixel (and only on the

floor pixels). \*/

glEnable(GL\_STENCIL\_TEST);

glStencilFunc(GL\_ALWAYS, 3, 0xffffffff);

glStencilOp(GL\_KEEP, GL\_KEEP, GL\_REPLACE);

}

}

/\* Draw "top" of floor. Use blending to blend in reflection. \*/

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

glColor4f(0.7, 0.0, 0.0, 0.3);

glColor4f(1.0, 1.0, 1.0, 0.3);

drawFloor();

glDisable(GL\_BLEND);

if (renderDinosaur) {

/\* Draw "actual" dinosaur, not its reflection. \*/

drawDinosaur();

}

if (renderShadow) {

/\* Render the projected shadow. \*/

if (stencilShadow) {

/\* Now, only render where stencil is set above 2 (ie, 3 where

the top floor is). Update stencil with 2 where the shadow

gets drawn so we don't redraw (and accidently reblend) the

shadow). \*/

glStencilFunc(GL\_LESS, 2, 0xffffffff); /\* draw if ==1 \*/

glStencilOp(GL\_REPLACE, GL\_REPLACE, GL\_REPLACE);

}

/\* To eliminate depth buffer artifacts, we use polygon offset

to raise the depth of the projected shadow slightly so

that it does not depth buffer alias with the floor. \*/

if (offsetShadow) {

switch (polygonOffsetVersion) {

case EXTENSION:

#ifdef GL\_EXT\_polygon\_offset

glEnable(GL\_POLYGON\_OFFSET\_EXT);

break;

#endif

#ifdef GL\_VERSION\_1\_1

case ONE\_DOT\_ONE:

glEnable(GL\_POLYGON\_OFFSET\_FILL);

break;

#endif

case MISSING:

/\* Oh well. \*/

break;

}

}

/\* Render 50% black shadow color on top of whatever the

floor appareance is. \*/

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

glDisable(GL\_LIGHTING); /\* Force the 50% black. \*/

glColor4f(0.0, 0.0, 0.0, 0.5);

glPushMatrix();

/\* Project the shadow. \*/

glMultMatrixf((GLfloat \*) floorShadow);

drawDinosaur();

glPopMatrix();

glDisable(GL\_BLEND);

glEnable(GL\_LIGHTING);

if (offsetShadow) {

switch (polygonOffsetVersion) {

#ifdef GL\_EXT\_polygon\_offset

case EXTENSION:

glDisable(GL\_POLYGON\_OFFSET\_EXT);

break;

#endif

#ifdef GL\_VERSION\_1\_1

case ONE\_DOT\_ONE:

glDisable(GL\_POLYGON\_OFFSET\_FILL);

break;

#endif

case MISSING:

/\* Oh well. \*/

break;

}

}

if (stencilShadow) {

glDisable(GL\_STENCIL\_TEST);

}

}

glPushMatrix();

glDisable(GL\_LIGHTING);

glColor3f(1.0, 1.0, 0.0);

if (directionalLight) {

/\* Draw an arrowhead. \*/

glDisable(GL\_CULL\_FACE);

glTranslatef(lightPosition[0], lightPosition[1], lightPosition[2]);

glRotatef(lightAngle \* -180.0 / M\_PI, 0, 1, 0);

glRotatef(atan(lightHeight/12) \* 180.0 / M\_PI, 0, 0, 1);

glBegin(GL\_TRIANGLE\_FAN);

glVertex3f(0, 0, 0);

glVertex3f(2, 1, 1);

glVertex3f(2, -1, 1);

glVertex3f(2, -1, -1);

glVertex3f(2, 1, -1);

glVertex3f(2, 1, 1);

glEnd();

/\* Draw a white line from light direction. \*/

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINES);

glVertex3f(0, 0, 0);

glVertex3f(5, 0, 0);

glEnd();

glEnable(GL\_CULL\_FACE);

} else {

/\* Draw a yellow ball at the light source. \*/

glTranslatef(lightPosition[0], lightPosition[1], lightPosition[2]);

glutSolidSphere(1.0, 5, 5);

}

glEnable(GL\_LIGHTING);

glPopMatrix();

glPopMatrix();

if (reportSpeed) {

glFinish();

end = glutGet(GLUT\_ELAPSED\_TIME);

printf("Speed %.3g frames/sec (%d ms)\n", 1000.0/(end-start), end-start);

}

glutSwapBuffers();

}

/\* ARGSUSED2 \*/

static void

mouse(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON) {

if (state == GLUT\_DOWN) {

moving = 1;

startx = x;

starty = y;

}

if (state == GLUT\_UP) {

moving = 0;

}

}

if (button == GLUT\_MIDDLE\_BUTTON) {

if (state == GLUT\_DOWN) {

lightMoving = 1;

lightStartX = x;

lightStartY = y;

}

if (state == GLUT\_UP) {

lightMoving = 0;

}

}

}

/\* ARGSUSED1 \*/

static void

motion(int x, int y)

{

if (moving) {

angle = angle + (x - startx);

angle2 = angle2 + (y - starty);

startx = x;

starty = y;

glutPostRedisplay();

}

if (lightMoving) {

lightAngle += (x - lightStartX)/40.0;

lightHeight += (lightStartY - y)/20.0;

lightStartX = x;

lightStartY = y;

glutPostRedisplay();

}

}

/\* Advance time varying state when idle callback registered. \*/

static void

idle(void)

{

static float time = 0.0;

time = glutGet(GLUT\_ELAPSED\_TIME) / 500.0;

jump = 4.0 \* fabs(sin(time)\*0.5);

if (!lightMoving) {

lightAngle += 0.03;

}

glutPostRedisplay();

}

enum {

M\_NONE, M\_MOTION, M\_LIGHT, M\_TEXTURE, M\_SHADOWS, M\_REFLECTION, M\_DINOSAUR,

M\_STENCIL\_REFLECTION, M\_STENCIL\_SHADOW, M\_OFFSET\_SHADOW,

M\_POSITIONAL, M\_DIRECTIONAL, M\_PERFORMANCE

};

static void

controlLights(int value)

{

switch (value) {

case M\_NONE:

return;

case M\_MOTION:

animation = 1 - animation;

if (animation) {

glutIdleFunc(idle);

} else {

glutIdleFunc(NULL);

}

break;

case M\_LIGHT:

lightSwitch = !lightSwitch;

if (lightSwitch) {

glEnable(GL\_LIGHT0);

} else {

glDisable(GL\_LIGHT0);

}

break;

case M\_TEXTURE:

useTexture = !useTexture;

break;

case M\_SHADOWS:

renderShadow = 1 - renderShadow;

break;

case M\_REFLECTION:

renderReflection = 1 - renderReflection;

break;

case M\_DINOSAUR:

renderDinosaur = 1 - renderDinosaur;

break;

case M\_STENCIL\_REFLECTION:

stencilReflection = 1 - stencilReflection;

break;

case M\_STENCIL\_SHADOW:

stencilShadow = 1 - stencilShadow;

break;

case M\_OFFSET\_SHADOW:

offsetShadow = 1 - offsetShadow;

break;

case M\_POSITIONAL:

directionalLight = 0;

break;

case M\_DIRECTIONAL:

directionalLight = 1;

break;

case M\_PERFORMANCE:

reportSpeed = 1 - reportSpeed;

break;

}

glutPostRedisplay();

}

/\* When not visible, stop animating. Restart when visible again. \*/

static void

visible(int vis)

{

if (vis == GLUT\_VISIBLE) {

if (animation)

glutIdleFunc(idle);

} else {

if (!animation)

glutIdleFunc(NULL);

}

}

/\* Press any key to redraw; good when motion stopped and

performance reporting on. \*/

/\* ARGSUSED \*/

static void

key(unsigned char c, int x, int y)

{

if (c == 27) {

exit(0); /\* IRIS GLism, Escape quits. \*/

}

glutPostRedisplay();

}

/\* Press any key to redraw; good when motion stopped and

performance reporting on. \*/

/\* ARGSUSED \*/

static void

special(int k, int x, int y)

{

glutPostRedisplay();

}

static int

supportsOneDotOne(void)

{

const char \*version;

int major, minor;

version = (char \*) glGetString(GL\_VERSION);

if (sscanf(version, "%d.%d", &major, &minor) == 2)

return major >= 1 && minor >= 1;

return 0; /\* OpenGL version string malformed! \*/

}

int

main(int argc, char \*\*argv)

{

int i;

glutInit(&argc, argv);

for (i=1; i<argc; i++) {

if (!strcmp("-linear", argv[i])) {

linearFiltering = 1;

} else if (!strcmp("-mipmap", argv[i])) {

useMipmaps = 1;

} else if (!strcmp("-ext", argv[i])) {

forceExtension = 1;

}

}

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE | GLUT\_DEPTH | GLUT\_STENCIL | GLUT\_MULTISAMPLE);

#if 1

/\* In GLUT 4.0, you'll be able to do this an be sure to

get 2 bits of stencil if the machine has it for you. \*/

glutInitDisplayString("samples stencil>=2 rgb double depth");

#endif

glutCreateWindow("Shadowy Leapin' Lizards");

if (glutGet(GLUT\_WINDOW\_STENCIL\_SIZE) <= 1) {

printf("dinoshade: Sorry, I need at least 2 bits of stencil.\n");

exit(1);

}

/\* Register GLUT callbacks. \*/

glutDisplayFunc(redraw);

glutMouseFunc(mouse);

glutMotionFunc(motion);

glutVisibilityFunc(visible);

glutKeyboardFunc(key);

glutSpecialFunc(special);

glutCreateMenu(controlLights);

glutAddMenuEntry("Toggle motion", M\_MOTION);

glutAddMenuEntry("-----------------------", M\_NONE);

glutAddMenuEntry("Toggle light", M\_LIGHT);

glutAddMenuEntry("Toggle texture", M\_TEXTURE);

glutAddMenuEntry("Toggle shadows", M\_SHADOWS);

glutAddMenuEntry("Toggle reflection", M\_REFLECTION);

glutAddMenuEntry("Toggle dinosaur", M\_DINOSAUR);

glutAddMenuEntry("-----------------------", M\_NONE);

glutAddMenuEntry("Toggle reflection stenciling", M\_STENCIL\_REFLECTION);

glutAddMenuEntry("Toggle shadow stenciling", M\_STENCIL\_SHADOW);

glutAddMenuEntry("Toggle shadow offset", M\_OFFSET\_SHADOW);

glutAddMenuEntry("----------------------", M\_NONE);

glutAddMenuEntry("Positional light", M\_POSITIONAL);

glutAddMenuEntry("Directional light", M\_DIRECTIONAL);

glutAddMenuEntry("-----------------------", M\_NONE);

glutAddMenuEntry("Toggle performance", M\_PERFORMANCE);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

makeDinosaur();

#ifdef GL\_VERSION\_1\_1

if (supportsOneDotOne() && !forceExtension) {

polygonOffsetVersion = ONE\_DOT\_ONE;

glPolygonOffset(-2.0, -1.0);

} else

#endif

{

#ifdef GL\_EXT\_polygon\_offset

/\* check for the polygon offset extension \*/

if (glutExtensionSupported("GL\_EXT\_polygon\_offset")) {

polygonOffsetVersion = EXTENSION;

glPolygonOffsetEXT(-0.1, -0.002);

} else

#endif

{

polygonOffsetVersion = MISSING;

printf("\ndinoshine: Missing polygon offset.\n");

printf(" Expect shadow depth aliasing artifacts.\n\n");

}

}

glEnable(GL\_CULL\_FACE);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_TEXTURE\_2D);

glLineWidth(3.0);

glMatrixMode(GL\_PROJECTION);

gluPerspective( /\* field of view in degree \*/ 40.0,

/\* aspect ratio \*/ 1.0,

/\* Z near \*/ 20.0, /\* Z far \*/ 100.0);

glMatrixMode(GL\_MODELVIEW);

gluLookAt(0.0, 8.0, 60.0, /\* eye is at (0,8,60) \*/

0.0, 8.0, 0.0, /\* center is at (0,8,0) \*/

0.0, 1.0, 0.); /\* up is in postivie Y direction \*/

glLightModeli(GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, 1);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightColor);

glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.1);

glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.05);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

makeFloorTexture();

/\* Setup floor plane for projected shadow calculations. \*/

findPlane(floorPlane, floorVertices[1], floorVertices[2], floorVertices[3]);

glutMainLoop();

return 0; /\* ANSI C requires main to return int. \*/

}

.\examples\file3.c:

#!../c -Wall -std=c89 -pedantic --

#define \_DEFAULT\_SOURCE

#include <stdio.h>

#include <netdb.h>

#include <stdlib.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#include <sys/socket.h>

/\* http://linux.die.net/man/3/hstrerror \*/

void \_hstrerror(int err) {

switch (err) {

case HOST\_NOT\_FOUND:

fputs("error: host is unknown", stderr);

break;

case NO\_DATA:

fputs("error: name is valid but has no IP address", stderr);

break;

case NO\_RECOVERY:

fputs("error: nonrecoverable name server error occurred", stderr);

break;

case TRY\_AGAIN:

fputs("error: temporary error occurred, try again later", stderr);

break;

}

}

int main(int argc, char \*\*argv) {

struct hostent \*h;

struct in\_addr\*\*ia;

if (argc != 2) {

fprintf(stderr, "Usage: %s <host>\n", argv[0]);

return 1;

}

h = gethostbyname(argv[1]);

if (h) {

ia = (struct in\_addr\*\*)h->h\_addr\_list;

while (\*ia) printf("%s\n", inet\_ntoa(\*\*(ia++)));

} else {

\_hstrerror(h\_errno);

}

return 0;

}

.\examples\file4.c:

#!../c -ljansson -lcurl --

/\*

\* Copyright (c) 2009-2014 Petri Lehtinen <petri@digip.org>

\*

\* Jansson is free software; you can redistribute it and/or modify

\* it under the terms of the MIT license. See LICENSE for details.

\*/

#include <stdlib.h>

#include <string.h>

#include <jansson.h>

#include <curl/curl.h>

#define BUFFER\_SIZE (256 \* 1024) /\* 256 KB \*/

#define URL\_FORMAT "https://api.github.com/repos/%s/%s/commits"

#define URL\_SIZE 256

/\* Return the offset of the first newline in text or the length of

text if there's no newline \*/

static int newline\_offset(const char \*text)

{

const char \*newline = strchr(text, '\n');

if(!newline)

return strlen(text);

else

return (int)(newline - text);

}

struct write\_result

{

char \*data;

int pos;

};

static size\_t write\_response(void \*ptr, size\_t size, size\_t nmemb, void \*stream)

{

struct write\_result \*result = (struct write\_result \*)stream;

if(result->pos + size \* nmemb >= BUFFER\_SIZE - 1)

{

fprintf(stderr, "error: too small buffer\n");

return 0;

}

memcpy(result->data + result->pos, ptr, size \* nmemb);

result->pos += size \* nmemb;

return size \* nmemb;

}

static char \*request(const char \*url)

{

CURL \*curl = NULL;

CURLcode status;

struct curl\_slist \*headers = NULL;

char \*data = NULL;

long code;

curl\_global\_init(CURL\_GLOBAL\_ALL);

curl = curl\_easy\_init();

if(!curl)

goto error;

data = malloc(BUFFER\_SIZE);

if(!data)

goto error;

struct write\_result write\_result = {

.data = data,

.pos = 0

};

curl\_easy\_setopt(curl, CURLOPT\_URL, url);

/\* GitHub commits API v3 requires a User-Agent header \*/

headers = curl\_slist\_append(headers, "User-Agent: Jansson-Tutorial");

curl\_easy\_setopt(curl, CURLOPT\_HTTPHEADER, headers);

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_response);

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, &write\_result);

status = curl\_easy\_perform(curl);

if(status != 0)

{

fprintf(stderr, "error: unable to request data from %s:\n", url);

fprintf(stderr, "%s\n", curl\_easy\_strerror(status));

goto error;

}

curl\_easy\_getinfo(curl, CURLINFO\_RESPONSE\_CODE, &code);

if(code != 200)

{

fprintf(stderr, "error: server responded with code %ld\n", code);

goto error;

}

curl\_easy\_cleanup(curl);

curl\_slist\_free\_all(headers);

curl\_global\_cleanup();

/\* zero-terminate the result \*/

data[write\_result.pos] = '\0';

return data;

error:

if(data)

free(data);

if(curl)

curl\_easy\_cleanup(curl);

if(headers)

curl\_slist\_free\_all(headers);

curl\_global\_cleanup();

return NULL;

}

int main(int argc, char \*argv[])

{

size\_t i;

char \*text;

char url[URL\_SIZE];

json\_t \*root;

json\_error\_t error;

if(argc != 3)

{

fprintf(stderr, "usage: %s USER REPOSITORY\n\n", argv[0]);

fprintf(stderr, "List commits at USER's REPOSITORY.\n\n");

return 2;

}

snprintf(url, URL\_SIZE, URL\_FORMAT, argv[1], argv[2]);

text = request(url);

if(!text)

return 1;

root = json\_loads(text, 0, &error);

free(text);

if(!root)

{

fprintf(stderr, "error: on line %d: %s\n", error.line, error.text);

return 1;

}

if(!json\_is\_array(root))

{

fprintf(stderr, "error: root is not an array\n");

json\_decref(root);

return 1;

}

for(i = 0; i < json\_array\_size(root); i++)

{

json\_t \*data, \*sha, \*commit, \*message;

const char \*message\_text;

data = json\_array\_get(root, i);

if(!json\_is\_object(data))

{

fprintf(stderr, "error: commit data %d is not an object\n", (int)(i + 1));

json\_decref(root);

return 1;

}

sha = json\_object\_get(data, "sha");

if(!json\_is\_string(sha))

{

fprintf(stderr, "error: commit %d: sha is not a string\n", (int)(i + 1));

return 1;

}

commit = json\_object\_get(data, "commit");

if(!json\_is\_object(commit))

{

fprintf(stderr, "error: commit %d: commit is not an object\n", (int)(i + 1));

json\_decref(root);

return 1;

}

message = json\_object\_get(commit, "message");

if(!json\_is\_string(message))

{

fprintf(stderr, "error: commit %d: message is not a string\n", (int)(i + 1));

json\_decref(root);

return 1;

}

message\_text = json\_string\_value(message);

printf("%.8s %.\*s\n",

json\_string\_value(sha),

newline\_offset(message\_text),

message\_text);

}

json\_decref(root);

return 0;

}

.\examples\file5.c:

#!../c

#include <stdio.h>

int main(void) {

puts("Hello World!");

return 0;

}

.\examples\file6.c:

#!../c -std=c99 -Wall -Werror -DRUN --

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

int main(void) {

puts("Generating 10 random numbers...");

srand(time(NULL));

#ifdef RUN

for (int i = 0; i < 10; i++)

printf("%d\n", rand());

return 0;

#else

printf("error: compile with flag -DRUN\n");

return 1;

#endif

}

.\files\test:

.\views\index.php:

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport"

content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

</head>

<body>

<a href="/">Вернуться на главную</a>

<h1>

15. Пройти по дереву каталогов, начиная с текущего, и удалить все

файлы на языке Си, содержащие внутри максимальное количество

операторов if-else не менее 2, и имеющие шаблон имени файла: в названии

файла имеются две точки и дата в виде YYMMDD в любом месте и младше

месяца и старше недели с правами только на чтение.

</h1>

<div>

<form action="../controllers/generate.php">

<button type="submit">Сгенерировать случайные файлы</button>

</form>

<br>

</div>

<div>

<form action="../controllers/delete.php">

<button type="submit">Удалить файлы удовлетворяющие условию</button>

</form>

<br>

</div>

<div>

<?php include('./show.php'); ?>

</div>

</body>

</html>

.\views\show.php:

<?php

require\_once('../autoload.php');

use lab7\CatalogManager;

$catalogManager = new CatalogManager();

echo $catalogManager->listFolders("../files");