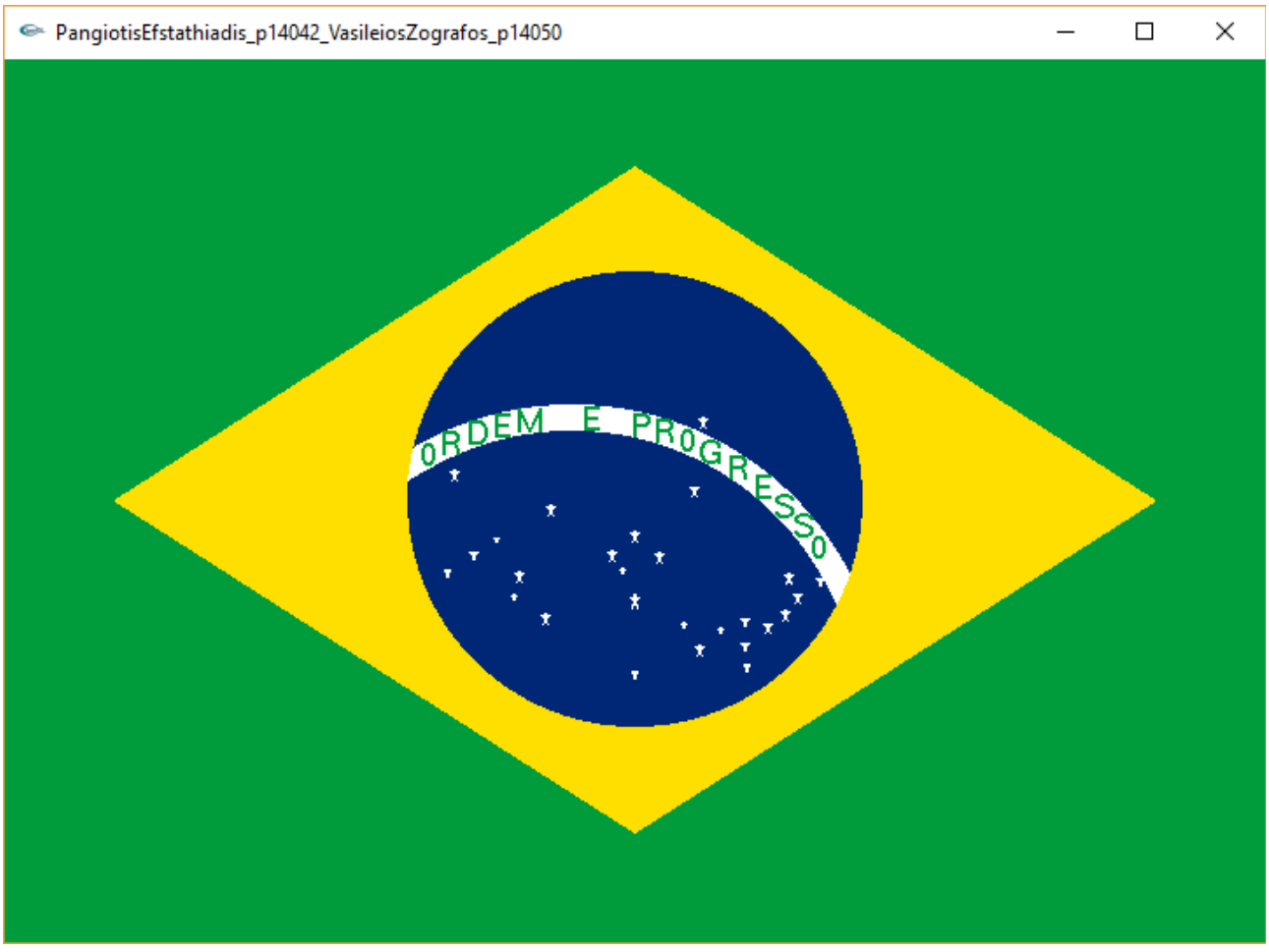
Εργασία Γραφικών Υπολογιστών

Παναγιώτης Ευσταθιάδης

<u>Θέμα 1ο</u>: *01_Brazil.png* (η παρακάτω εικόνα είναι screenshot εκτέλεσης κώδικα)



```
#include <Windows.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

typedef enum {
   MODE_BITMAP,
   MODE_STROKE
} mode_type;

static mode_type mode;
static int font_index;
```

Εισάγουμε τις **c++ libraries** που θα χρησιμοιήσουμε για το **θέμα 1**.
Ορίζουμε το **enum mode_type** και τις μεταβλητές **mode** και **font_index** τα οποία χρησιμοποιούμε για το **stroke** και το **bitmap** των γραμμάτων.

```
// this function draws a star
void DrawStar(float fX, float fY,float kfRadius) {
  float kfPi = 3.14159265358979323846264338327795;
  // Draw ten triangles
  float kfInnerRadius = kfRadius*(1.0/(sin((2.0*kfPi)/5.0)*2.0*cos(kfPi/10.0) + sin((3.0*kfPi)/10.0)));
  glColor3f(1.0, 1.0, 1.0);
  glBegin(GL_TRIANGLE_FAN);
    glVertex3f(fX, fY, 0.0);
    for (int iVertIndex = 0; iVertIndex < 10; ++iVertIndex) {
      float fAngleStart = kfPi/2.0 + (iVertIndex*2.0*kfPi)/10.0;
      float fAngleEnd   = fAngleStart + kfPi/5.0;
      if (iVertIndex % 2 == 0) {
        glVertex3f(fX + kfRadius*cos(fAngleStart)/1.9, fY + kfRadius*sin(fAngleStart), 0.0);
        glVertex3f(fX + kfInnerRadius*cos(fAngleEnd)/1.9, fY + kfInnerRadius*sin(fAngleEnd), 0.0);
      } else {
        glVertex3f(fX + kfInnerRadius*cos(fAngleStart)/1.9, fY + kfInnerRadius*sin(fAngleStart), 0.0);
        glVertex3f(fX + kfRadius*cos(fAngleEnd)/1.9, fY + kfRadius*sin(fAngleEnd), 0.0);
      }
    }
  glEnd();
}


void DrawC(){
  glColor3f(254.0/255.0,223.0/255.0,0.0/255.0);
  glBegin(GL_QUADS);
    glVertex2f(62.0,252.0);
    glVertex2f(360.0,62.0);
    glVertex2f(658.0,252.0);
    glVertex2f(360.0,443.0);
  glEnd();
}
```

Ορίζουμε εδώ τις συναρτήσεις:
drawStar και **drawC**.
Η **drawStar** χρησιμοποιείται για την δημιουργία αστερού.
Η **drawC** χρησιμοποιείται για την δημιουργία ρόμβου.

```
void drawFilledSun(float x, float y, float radius, int num_segments){
  float i;
  double twicePi = 2.0 * 3.142;
  glBegin(GL_TRIANGLE_FAN);
  glVertex2f(x, y);
  for (i = 0; i <= num_segments; i++)   {
      glVertex2f (
      (x + (radius * cos(i * twicePi / num_segments))), (y + (radius * sin(i * twicePi / num_segments)))
      );
  }
  glEnd();
}
```

Η συνάρτηση **drawFilledSun** σχεδιάζει έναν γεμισμένο κύκλο.

```
void drawFilledArcSun(float x, float y, float radius, float theta_initial, float theta_final, int num_segments){

  float i;
  double twicePi = 2.0 * 3.142;
  glBegin(GL_TRIANGLE_FAN);
  glVertex2f(x, y);
  for (i = 0; i <= num_segments; i++)   {
      float theta = 2.0f * 3.1415926f * float(i) / float(num_segments);
      if (theta>=theta_initial*(2.0f * 3.1415926f)/360){
            if(theta<=theta_final*(2.0f * 3.1415926f)/360){
              glVertex2f (
                  (x + (radius * cos(i * twicePi / num_segments))), (y + (radius * sin(i * twicePi / num_segments)))
                );
            }}
  }

  glEnd();
  glColor3f(1,1,1);
  glBegin(GL_POLYGON);
  glVertex2f(233.1,504-222);
  glVertex2f(241,504-234.3);
  glVertex2f(229.8,504-241);
  glEnd();
  glBegin(GL_POLYGON);
  glVertex2f(482.7, 504-295.0);
  glVertex2f(470.0, 504-301.123);
  glVertex2f(475.5, 504-311.5);
  glEnd();
  glColor3f(0,155.0/255.0,58.0/255.0);
  glRasterPos3f( 237 , 504-232 ,0.0f );
  glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, '0');
  glColor3f(0,155.0/255.0,58.0/255.0);
  glRasterPos3f( 249 , 504.0-225.1 ,0.2f );
  glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'R');
  glColor3f(0,155.0/255.0,58.0/255.0);
```

Η συνάρτηση **drawFilledArcSun** σχεδίαζει
τα τόξα που χρησιμοποιούνται για τον σχηματισμό της λευκής λορίδας που υπάρχει στην σημαία της Βραζιλίας.

```
glRasterPos3f( 264 , 504-220 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'D');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 279 , 504.0-215.5 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'E');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 292 , 504-213 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'M');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 330 , 504-212 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'E');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 358 , 504-216 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'P');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 371 , 504-219 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'R');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 385 , 504-224 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, '0');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 396 , 504-231 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'G');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 413 , 504-240 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'R');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 428 , 504-251 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'E');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 439 , 504-262 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'S');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 450 , 504-273 ,0.2f );
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, 'S');
glColor3f(0,155.0/255.0,58.0/255.0);
glRasterPos3f( 460 , 504-285 ,0.2f );
```

```
  glRasterPos3f( 460 , 504-285 ,0.2f );
  glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, '0');

}
```

Επίσης, είναι υπεύθηνη για την δημιουργία των γραμμάτων πάνω στην λευκή λορίδα της σημαίας.

```
void DrawStars() {
DrawStar(253,211,5.0);
DrawStar(399,297,6.1);
DrawStar(257,267,5.8);
DrawStar(268,221,5.1);
DrawStar(281,230,4.3);
DrawStar(294,209,5.666);
DrawStar(291,197,4.6);
DrawStar(309,185,5.7);
DrawStar(409,178,4.5);
DrawStar(423,169,5.2);
DrawStar(424,157,5.0);
DrawStar(423,183,5.1);
DrawStar(436,180,5.3);
DrawStar(446,187,5.9);
DrawStar(453,197,5.3);
DrawStar(448,208,6.3);
DrawStar(466,206,5.2);
DrawStar(394,258,5.3);
DrawStar(360,232,5.5);
DrawStar(312,247,5.9);
DrawStar(347,221,5.6);
DrawStar(353,212,4.61);
DrawStar(360,195,6.5);
DrawStar(374,220,5.66);
DrawStar(360,153,4.95);
DrawStar(388,181,4.7);
DrawStar(397,167,5.9);
}
```

Η συνάτηση **DrawStars** σχεδίαζει όλα τα αστέρια
της σημαίας καλώντας την συνάρτηση **DrawStar**.

```
void Draw() {
  glClear(GL_COLOR_BUFFER_BIT);
  DrawC();
  glColor3f(0.0/255.0, 39.0/255.0, 118.0/255.0);

    drawFilledSun(360,253,130,169);
    //drawFilledArcSun(0.5,0.5,0.175,0.0,360.0,1331);
  glColor3f(1,1,1);
  //DrawArcCircle(298,50,246,41,105,366666);
//  DrawArcCircle(298,50,236,39.5,106.3,366666);
  drawFilledArcSun(324,128,179,26.5,121.5,400);
  glColor3f(0.0/255.0, 39.0/255.0, 118.0/255.0);
  drawFilledArcSun(324,128,164,26.5,121.5,400);
  glColor3f(1,1,1);
  DrawStars();
  glEnd();
  glFlush();
  glutSwapBuffers();

}
```

Η συνάρτηση **Draw** σχεδιάζει την σημαία της Βραζιλίας.

```
void Initialize() {
  glClearColor(0.0/255.0, 155.0/255.0, 58.0/255.0, 0.0);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0, 720, 0, 504, -1, 1);
  mode = MODE_BITMAP;
   font_index = 0;

}

int main(int iArgc, char** cppArgv) {
  glutInit(&iArgc, cppArgv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(720, 504);
  glutInitWindowPosition(100, 100);
  glutCreateWindow("PangiotisEfstathiadis_p14042_VasileiosZografos_p14050");
  Initialize();
  glutDisplayFunc(Draw);
  //glutMouseFunc(onMouseClick);

  glutMainLoop();
  return 1;
}
```

Η συνάρτηση **Initialize** είναι υπεύθυνη για την δημιουργία πράσινου φόντου και για τις ρυθμίσεις σχετικά με τα γράμματα που θα σχεδιαστούν πάνω στην σημαία.
Η συνάρτηση **main** δημιουργεί το παράθυρο στο οποίο θα σχεδιαστεί η σημαία.

Θέμα 2ο:

Video: 07_House On Little Cubes by Kunio Katô

```
#include <Windows.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <unistd.h>
#include <time.h>

#define MAX_PARTICLES 234
float slowdown = 11.0;
float velocity = 0.0;
int loop;
int fall;

typedef struct {
  bool alive;
  float life;
  float fade;

  float red;
  float green;
  float blue;

  float xpos;
  float ypos;

  float vel;

  float gravity;
}particles;

// Paticle System
particles par_sys[MAX_PARTICLES];
```

Εισάγουμε τις απαραίτητες βιβλιοθήκες για την 2η άσκηση.
Επίσης, δημιουργούμε τις απαραίτητα στοιχεία (μεταβλητές **slowdown**, **velocity**, **loop**, **fail**, η σταθερά **MAX_PARTICLES**, ο πίνακας **par_sys** και το struct **particles**) για να δημιουγίσουμε το particle system το οποίο θα δημιουργεί το animation της βροχής.

```
// Reset Particles
void initParticles(int i) {
    par_sys[i].alive = true;
    par_sys[i].life = 40.0;
    par_sys[i].fade = float(rand()%100)/1000.0f+0.003f;

    //par_sys[i].xpos = (float) (rand() % 21) - 10;
    par_sys[i].xpos = (float) (rand() % 720) - 10;
    par_sys[i].ypos = 480.0;

    par_sys[i].red = 1.0;
    par_sys[i].green = 1.0;
    par_sys[i].blue = 1.0;

    par_sys[i].vel = velocity;
    par_sys[i].gravity = -0.2;//-0.8;

}

GLfloat xRotated, yRotated, zRotated, roloi, xb1, yb1, circle;
void init(void)
{
  glClearColor(80.0/255.0,45.0/255.0,28.0/255.0,0);

     // Initialize particles
    for (loop = 0; loop < MAX_PARTICLES; loop++) {
        initParticles(loop);
    }

}
```

Η συνάρτηση **initParticles** είναι υπεύθυνη για την δημιουργία και την επανοφορά των σωματιδίων.

Η συνάρτηση **init** συμβάλλει στην αρχικοποίηση του particle system.

```
void drawRain() {
  float x, y, paxos;


  for (loop = 0; loop < MAX_PARTICLES; loop=loop+2) {
    if (par_sys[loop].alive == true) {
      x = par_sys[loop].xpos;
      y = par_sys[loop].ypos;
      glColor3f(1.0, 1.0, 1.0);

    paxos=(float) (rand() % 15);
      glColor3f(1, 1, 1);
        glBegin(GL_LINE_STRIP);
          glVertex3f (x+3.51,y+paxos*2.7,1);
          glVertex3f (x+4.52,y+paxos*3.3,1);
          glVertex3f (x+5.01,y+paxos*4.1,1);
          glVertex3f (x+5.61,y+paxos*5.2,1);
      glEnd();

  glPushMatrix();
  glTranslatef(x, y,44);

glPopMatrix();
      par_sys[loop].ypos += par_sys[loop].vel / (slowdown*1000);
      par_sys[loop].vel += par_sys[loop].gravity;


      par_sys[loop].life -= par_sys[loop].fade;

      if (par_sys[loop].ypos <= -10) {
        int xi = x + 10;
        par_sys[loop].life = -1.0;
      }
      if (par_sys[loop].life < 0.0) {
        initParticles(loop);
      }
    }
  }
}
```

Η συνάρτηση **drawRain** ελέγχει το ***partcle-system*** της βροχής.

```
void animation(void){
  roloi+=1;
  DrawAnimation();
  glutPostRedisplay();

}

void reshape(int x, int y){
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0, 720.0, 0.0, 480.0, -1.0,1.0);
    glMatrixMode(GL_MODELVIEW);
    mode = MODE_STROKE;
   font_index = 0;
}

int main(int argc, char** argv){
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(720,480);
glutInitWindowPosition(0, 0);
glMatrixMode(GL_PROJECTION);
glOrtho(0, 720.0, 0.0, 480.0, -1.0,1.0);
glutCreateWindow(argv[0]);
init();
glutDisplayFunc(DrawAnimation);
glutReshapeFunc(reshape);
glutIdleFunc(animation);
glutMainLoop();
return 0;
}
```

Εδώ ορίζουμε τις συνρτήσεις **animation**, **reshape** και **main**.
Η συνάρτηση **main** δημιουργεί το παράθυρο μέσα στο οποίο θα εκτελεστεί το animation και γράφουμε κάποιες ρυθμίσεις σχετικά με το animation και το OpenGL.
Η συνάρτηση **animation** ελέγχει το ρολοί και καλεί την συνάρτηση **DrawAnimation** η οποία, ελέγχει όλο το animation που θα βλέπει ο χρήστης (δηλαδή την βροχη, τον πρωταγωνιστή και το περιβάλλον).

Η συνάρτηση **reshape** είναι υπεύθυνη τον έλεγχο του περιεχομένου του παραθύρου σε περίπτωση αλλαγής των διαστάσεων του.

```
void omp_plus_Stuff(void)
```

Η **omp_plus_Stuff** δημιουργεί την ομπρέλα, τον ουρανό, τον κουβά, τους τοίχους, ένα αντικείμενο δίπλα από τον κουβά και τα πόδια του πρωταγωνιστή.

```
int initial_time=time(NULL), final_time,frame_count=0;
int fps=0;
```

Εδώ αρχικοποιούμε τις μεταβλητές **inintial_time** και **frame_count** και **fps** (frames per second) και ορίζουμε την μεταβλητή **final_time**. Αυτές οι μεταβλητές θα χρησιμεύσουν στην απεικόνηση των frames στο παράθυρο.

```
void output(GLfloat x, GLfloat y, char *text){
char *p;
glPushMatrix();
glTranslatef(x, y, 0);
for (p = text; *p; p++)
    glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
glPopMatrix();
}
```

Η συνάρτηση **output** εμφανίζει κείμενο στο παράθυρο.

```
using namespace std;
GLfloat xRotated, yRotated, zRotated, roloi, xb1, yb1, circle;
```

Εδώ έχουμε την χρήση του namespace std και τον ορισμό των μεταβλητών xRotated, yRorated, zRotated, roloi, xb1, yb1 για τον σχεδιασμό κύκλου και circle.

```
void DrawAnimation(void){

if (roloi>0*3600){
DrawAnimation0();drawRain();
}
  if (roloi>1*3600){
DrawAnimation2();drawRain();
}

if (roloi>2*3600){
DrawAnimation3();drawRain();
}

  if (roloi>3*3600){
DrawAnimation4();drawRain();
}
  if (roloi>4*3600){
DrawAnimation45();drawRain();
}
  if (roloi>5*3600){
DrawAnimation5();drawRain();
}
if (roloi>6*3600){
DrawAnimation6();drawRain();
}
if (roloi>7*3600){
DrawAnimation7();drawRain();
}
glFlush();
frame_count=0;
}
```

Η συνάρτηση **DrawAnimation** καλεί μέσα στο σώμα της τις συναρτήσεις **DrawAnimation0**, **DrawAnimation1**, **DrawAnimation3**, **DrawAnimation4**, DrawAnimation45, **DrawAnimation5**, **DrawAnimation6, DrawAnimation7** ανάλογα με την τιμή του ρολογιού.
Οι συναρτήσεις **DrawAnimation1**, **DrawAnimation3**, **DrawAnimation4**, DrawAnimation45, **DrawAnimation5** και **DrawAnimation6, DrawAnimation7** ευθύνοται για την κίνηση και δημιουγία του προσώπου του πρωταγωνιστή, του κορμού του των χεριών το τούβλο, το κεφάλι, την πίπα, τα μαλλιά.

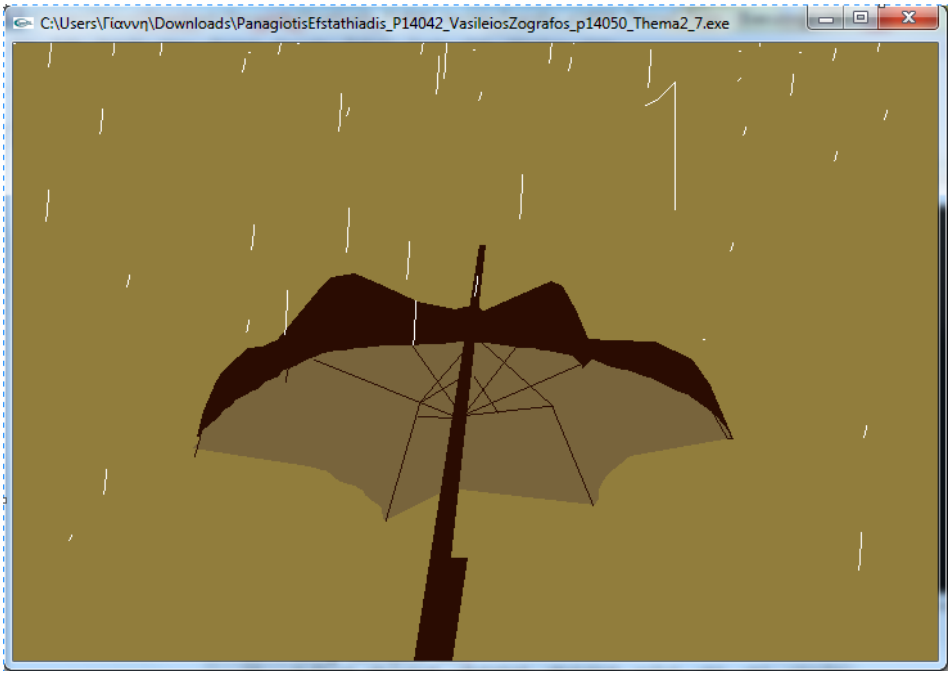
Η συνάρτηση **DrawAnimation0** δημιουργεί την βροχή και την ομπρέλα.

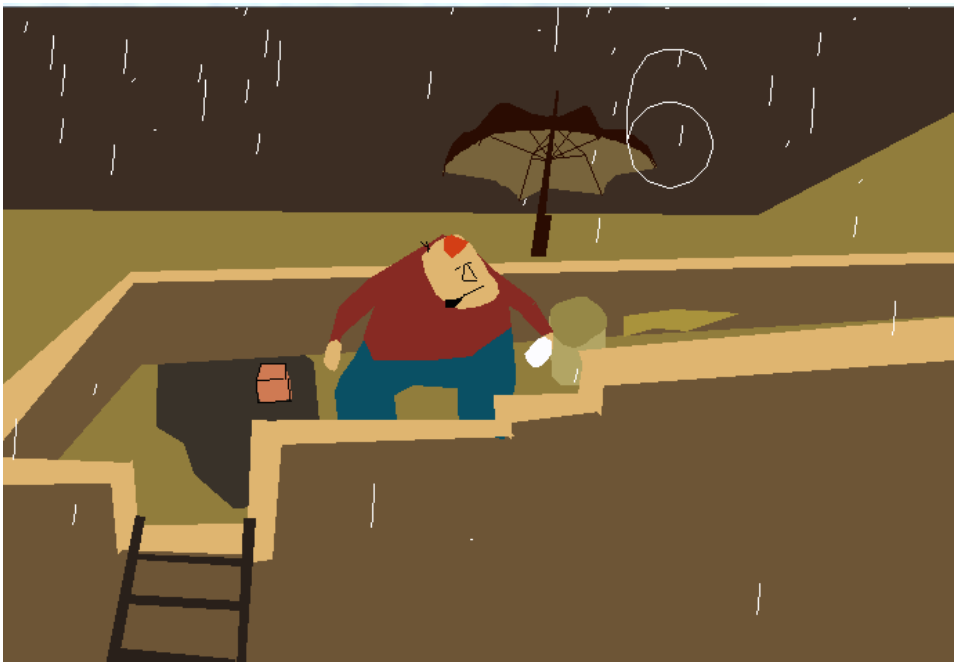
Το animation βασίστηκε στο διάστημα: **3:25-3:29** του βίντεο.

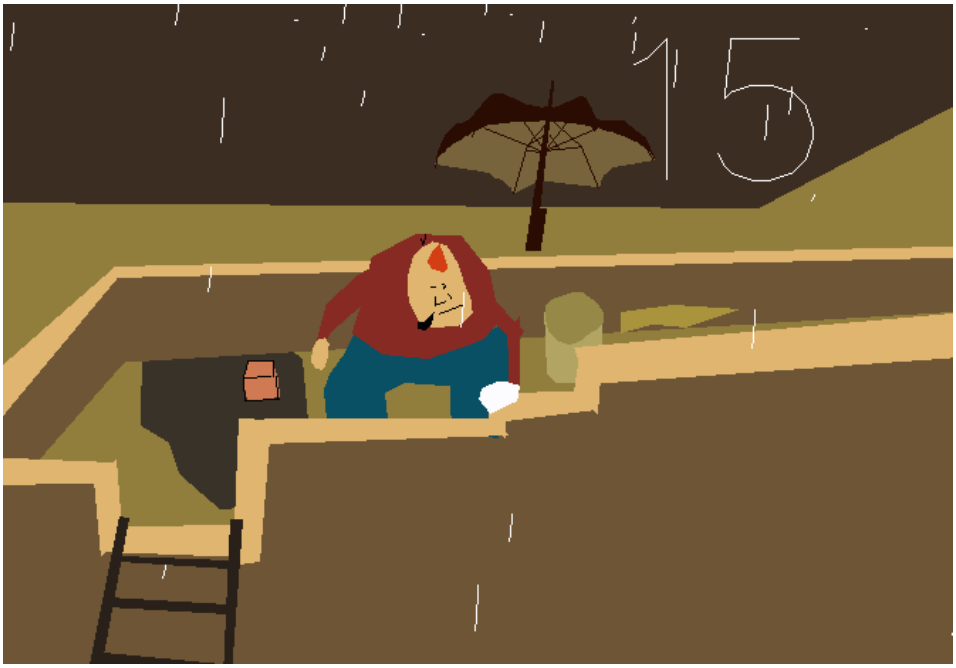
Παρακάτω έχουμε screen shots από το animation και την φωτογραφία της σημαίας που χρησιμοποιήσαμε:

Η παραπάνω εικόνα είναι η σημαία Βραζιλίας.

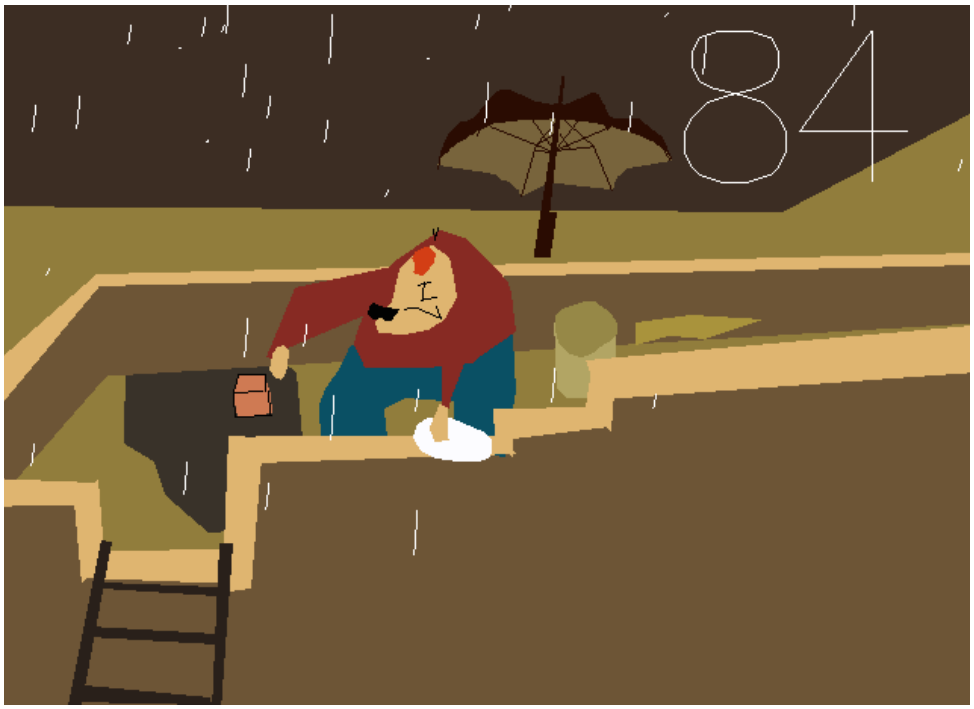
Τα screenshots από το animation:

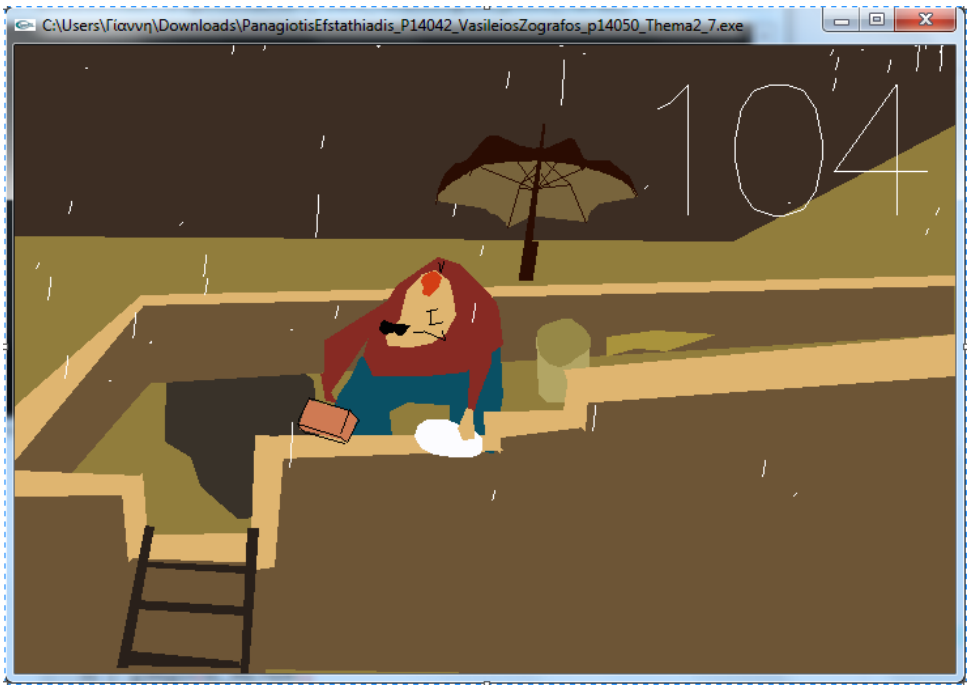



6


15


32


84


C:\Users\Γιαννη\Downloads\PanagiotisEfstathiadis_P14042_VasileiosZografos_p14050_Thema2_7.exe
104