Contents

1 Specification

The program shows the fable of The Wolf and the Kid through 4 scenes with one window of animation and another with the subtitles and sound. If the user clicks on the animation, then the animation and sound pauses. The animation starts off with a kid alone in a field with birds flying over him. It then transitions into the wolf and kid walking towards each other accompanied. Next, the Kid dances while the Wolf plays the pipe which can be heard. Then, there is an abandoned field with an animated cloud where the Dog chases the wolf into and soon jumps on the wolf allowing the Kid to escape. Then, we learn the moral of the animation.

2 Analysis

Input There is a button that is used for user input. The button takes place on the whole screen so the user will be able to click anywhere on the animation in order to pause and unpause the animation.

Process There is a function called pauseCB that desribes that the program automatically runs and when the user clicks on any part of the window, the movie will be paused. The pause is defined and called within the main.cpp file within this lab in order for the function to work. The process of drawing parts of the characters and animating the birds will be performed as individual functions are defined. There are 3 functions defined to put together the total drawings of the wolf, goat, and dog. These 3 functions work together in the drawCB2 file to put together the animation of the movie on the second window. The first window shows the subtitles of what is going on in the instant of the animation. The first scene introduces the goat and shows the goat in the field all alone. In the second scene, we meet the wolf who confronts the goat causing the goat to bargain for his life. In the third scene, we see that the wolf and goat came to an agreement and the deal was that they would dance together while the wolf pipes a tune before the wolf's big meal. The fourth scene shows that the dog heard the piping of the wolf and attacked the wolf so that the goat would be able to escape.

Output If paused, the screen will be still until the user unpauses the animation. The output will be the animation of the 4 scenes of the fable with the subtitles showing up on a window next to the animation. The first scene introduce the goat and shows the goat in the field all alone. In the second scene, we meet the wolf who confronts the goat causing the goat to bargain for his life. In the third scene, we see that the wolf and goat came to an agreement and the deal was that they would dance together while the wolf pipes a tune before the wolf's big meal. The fourth scene shows that the dog heard the piping of the wolf and attacked the wolf so that the goat would be able to escape. The moral of the story is "Do not let anything turn you from your purpose."

3 Design

- main.cpp was created as a main file to link all other functions into this and run together as an executable program. Then, the following functions were created.
- drawCB1.cpp This calls all of the functions in order for the first window to successfully have the subtitles. The first thing to do is set the background to a clear color like gray and then make the font something legible and a size big enough for anyone to read. Then, a file named readFable was created so it is called upon in this function in order to read the fable that comes from the text file that the fable is written in. Each scene is split into a separate array so the time changes per scene would change the correct words on the subtitles window. The times for each scene are defined in lab.h so that they only have to changed in one place
- drawCB2.cpp We start off the code with a few if statements that allow us to see that the backgrounds change according to what time it is. There are different backgrounds and that is seen by the different function names.Then, there is a scale command that will scale all of the vector characters that have been created to be in this animation. Once again, we manipulate if statements according to the timeline and when the time reaches a certain time, the scene will change into the next scene until the animation is over. Other than using a translate function, there is a special function called pts[timeline] that uses gnuplot data and points in order to create a Bezier curve in order to ensure that there is separate movement for all of the characters and so that all of the characters don't end up moving together.
- lab.h All information that needs to be repeated is put into file lab.h so that an include statemnt can be used for the information rather than retyping it several times in all of the different function files.
- drawHead.cpp This function is to draw the head of the wolf on the graphics window.
- drawEyes.cpp This function is to draw the eyes of the wolf on the graphics window.
- drawWolf.cpp This function is to draw the wolf on the graphics window using the commands in it.

- mirrorY.cpp This function is to mirror the image on Y coordinate.
- scaler.cpp Scale is the unit size used for the x and y coordinates and that is set to what we need for the characters.
- moveCharacter.cpp This is the translate part of the animation and will cause the characters to move.
- drawBackground.cpp This gives the second window a background of a pasture so that the characters and birds are moving on top of the background rather than a blank screen.
- drawBackground1.cpp This gives the second window a background of the second scene so that the characters are moving on top of the background rather than a blank screen.
- drawBackground2.cpp This gives the second window a background of the third scene so that the characters are moving on top of the background rather than a blank screen.
- drawBackground3.cpp This gives the second window a background of the last scene so that the characters and clouds are moving on top of the background rather than a blank screen.
- drawBirds.cpp This is the animation of the birds moving around at the top of the window.
- getBirds.cpp This allows us to see the GIF of the birds run in the first scene through an if statement to see if there is outupt in the array to run the animation.
- drawCloud.cpp This is the animation of the cloud moving around at the top of the window.
- drawBody.cpp The head of the wolf is drawn to the graphics window.
- drawdbody.cpp The body of the dog is drawn to the graphics window.
- drawdear1.cpp The left ear of the dog is drawn to the graphics window.
- drawdear2.cpp The right ear of the dog is drawn to the graphics window.

- drawdhead.cpp The dog's head is drawn to the graphics window.
- drawdleg1.cpp The dog's left leg is drawn to the graphics window.
- drawdleg2.cpp The dog's right leg is drawn to the graphics window.
- drawDog.cpp All of the functions to draw the dog are combined in this function to draw the total dog.
- drawGoat.cpp All of the functions to draw the goat are combined in this function to draw the total goat.
- drawLeg1.cpp The wolf's left leg is drawn to the graphics window.
- draw Leg2.cpp The wolf's right leg is drawn to the graphics window.
- drawNose.cpp The wolf's nose is drawn to the graphics window.
- drawTail.cpp The wolf's tail is drawn to the graphics window.
- gBody.cpp The goat's body is drawn to the graphics window.
- gEye.cpp The goat's eye is drawn to the graphics window.
- gHead.cpp The goat's head is drawn to the graphics window.
- gHorn1.cpp The goat's left horn is drawn to the graphics window.
- gHorn2.cpp The goat's right horn is drawn to the graphics window.
- gLeg1.cpp The goat's right leg is drawn to the graphics window.
- gLeg2.cpp The goat's left leg is drawn to the graphics window.
- gTail.cpp The goat's tail is drawn to the graphics window.

- fables.txt Our team fable is called from here rather than individually typing it out.
- readFable.cpp Our fable is read from a file rather than written directly on the screen.
- showText.cpp The text of our fable is shown on a screen that is right next to that of the animation and which is time synced.
- getPoints.cpp This function gets points from the data files that was written in gnuplot and the points would follow the bezier curve and so it is called here in order for each character to have its own path.
- getPoints2.cpp This function gets points2 from the data2 files that was written in gnuplot and the points2 would follow the bezier curve and so it is called here in order for each character to have its own path.
- getPoints3.cpp This function gets points3 from the data3 files that was written in gnuplot and the points3 would follow the bezier curve and so it is called here in order for each character to have its own path.
- soundEffects.cpp This allows the animation to have sound to enhance what the characters are doing. The sounds play along with the timeline and approximately when the scenes change to represent what is going on. The sound bits were uploaded to our directory in order for it to work.

4 Implementation

4.1 main.cpp

```
#include "lab.h"
```

```
Owner:Anusha
```

```
This function creates the initial window
Then sets a callback function so that the
graphics subsystem will use that function
to redraw the window whenever the OS
decides it needs to do that.
```

```
bool nopause; //this is a definition
int timeline = 0;
Fl_Cairo_Window cw1(WIDTH,HEIGHT);
Fl_Cairo_Window cw2(WIDTH,HEIGHT);
int main()
{

    Fl_Button b(0,0,WIDTH, HEIGHT);
```

```
The width is the whole size of the screen
so that the user can click anywhere they want
in order to pause the animation.
```

```
    b.callback(pauseCB);
    nopause = true;

        getPoints(); //function called so that characters can move
            separately
        getPoints2();
        getPoints3();


    getBirds(); //birds can be animated

    cw1.set_draw_cb(drawCB1);
    cw2.set_draw_cb(drawCB2);
    cw1.show(); //shows subtitles
   cw2.show(); //shows animation



Call this function in 1.0 seconds

    Fl::add_timeout(1, moveCharacter);
    Fl::run(); //passes control to FLTK
    return 0;
}
```

4.2 drawCB1.cpp

```
#include "lab.h"
#include <sstream>
```

```
This calls all of the functions in order for the first
window to successfully have the subtitles.  The first thing
to do is set the background to a clear color like gray
and then make the font something legible and a size
big enough for anyone to read.  Then, a file named
readFable was created so it is called upon in this function
in order to read the fable that comes from the text file
that the fable is written in.  Each scene is split into a separate
array so the time changes per scene would change the correct
words on the subtitles window.  The times for each scene are
defined in lab.h so that they only have to changed in one place.
```

```
This is what the subtitles looks like:
```

There was once a little Kid who thought he was strong because of his growing horns.

Owner: Tanmay

```
void drawCB1(Fl_Cairo_Window* cw,cairo_t* cr)
{
```

Set screen background

```
    cairo_set_source_rgb(cr, 0.1, 0.1, 0.1);
```

Set font type and size

```
cairo_select_font_face(cr, "Purisa",
                       CAIRO_FONT_SLANT_NORMAL,
                       CAIRO_FONT_WEIGHT_BOLD);

cairo_set_font_size(cr, 48);


std::string fable = readFable();

std::istringstream stream(fable);
std::string line;
int arrSize = 0;
while (std::getline(stream, line)) {
    arrSize++;
}

std::istringstream stream2(fable);
string fableArr[arrSize];
for (int i = 0; std::getline(stream2, line); i++) {
    fableArr[i] = line;
}

//std::cout << fable;

std::string sceneOneText = fableArr[0];
std::string sceneTwoText = fableArr[1];
std::string sceneThreeText = fableArr[2];
std::string sceneFourText = fableArr[3];
```

```
    if (timeline <= sceneOne) {
        showText(cr, sceneOneText);
    }
    else if (timeline > sceneOne && timeline <= sceneTwo) {
        showText(cr, sceneTwoText);
    }

    else if (timeline > sceneTwo && timeline <= sceneThree) {
        showText(cr, sceneThreeText);
    }
    else if (timeline > sceneFour) {
        showText(cr, sceneFourText);
    }


}
```

4.3 drawCB2.cpp

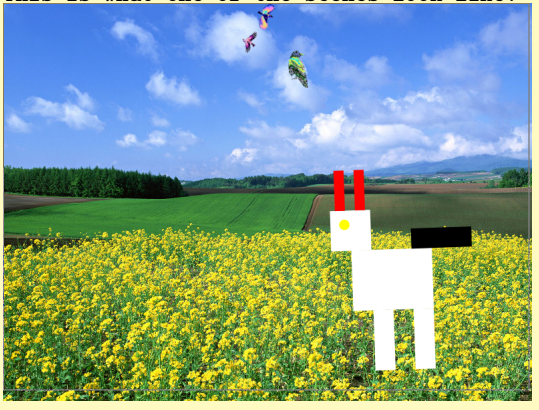
```
#include "lab.h"
```

```
Owner:Anusha
```

```
This function is to call all other drawing functions for the second window.
```

```
We start off the code with a few if statements
that allow us to see that the backgrounds change
according to what time it is.  There are different
backgrounds and that is seen by the different
function names.Then, there is a scale command
that will scale all of the vector characters that
have been created to be in this animation.
Once again, we manipulate if statements
according to the timeline and when the time
reaches a certain time, the scene will change into
the next scene until the animation is over.  Other
than using a translate function, there is a special
function called pts[timeline] that uses gnuplot
data and points in order to create a Bezier curve
in order to ensure that there is separate movement
for all of the characters and so that all of the
characters don't end up moving together.
```

This is what one of the scenes look like:



```
void drawCB2(Fl_Cairo_Window* cw,cairo_t* cr)
{

    if (timeline < sceneOne) {
        drawBackground(cr); //background for scene 1
        drawBirds(cr); //GIF for scene 1
    }

    if (timeline >= sceneOne && timeline <= sceneTwo) {
        drawBackground1(cr); //background for scene 2
```

```
    }

    if (timeline > sceneTwo && timeline<= sceneThree) {
        drawBackground2(cr);  //background for scene 3
    }

    if (timeline > sceneThree && timeline<=sceneFour) {
        drawBackground3(cr); //background for scene 4
        drawCloud(cr); //GIF for scene 4
    }
    if (timeline > sceneFour) { //extends the animation to keep running
        after the 4th scene until user chooses to stop using the pause
        button
        drawBackground3(cr);
        drawCloud(cr);
    }

    double x = 0.5;
    double y = 1;
    double sx = 0.5;
    double sy = 0.5;
```

apply scaler the value for ty is in pixels

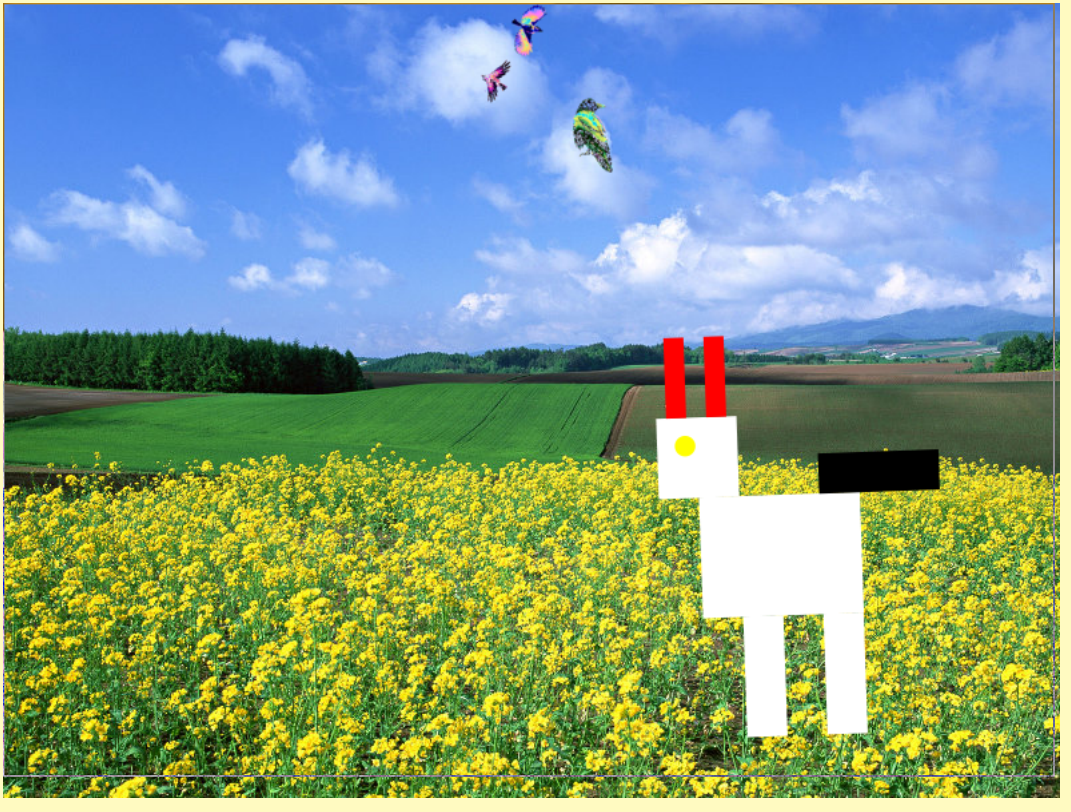
```
    static double tx = 0.0;
    static double ty = scaler(5.0);
    const double pixel = 1.0/WIDTH;
```

If statement that stops movement of the characters when it reaches a certain time

```
cairo_scale(cr,sx,sy); //scales characters throughout the whole
    animation
```

When the timeline is less than 10 seconds, the goat rotates to reach better food in the pasture.

The animation starts with scene 1 where there is a goat alone on the field.

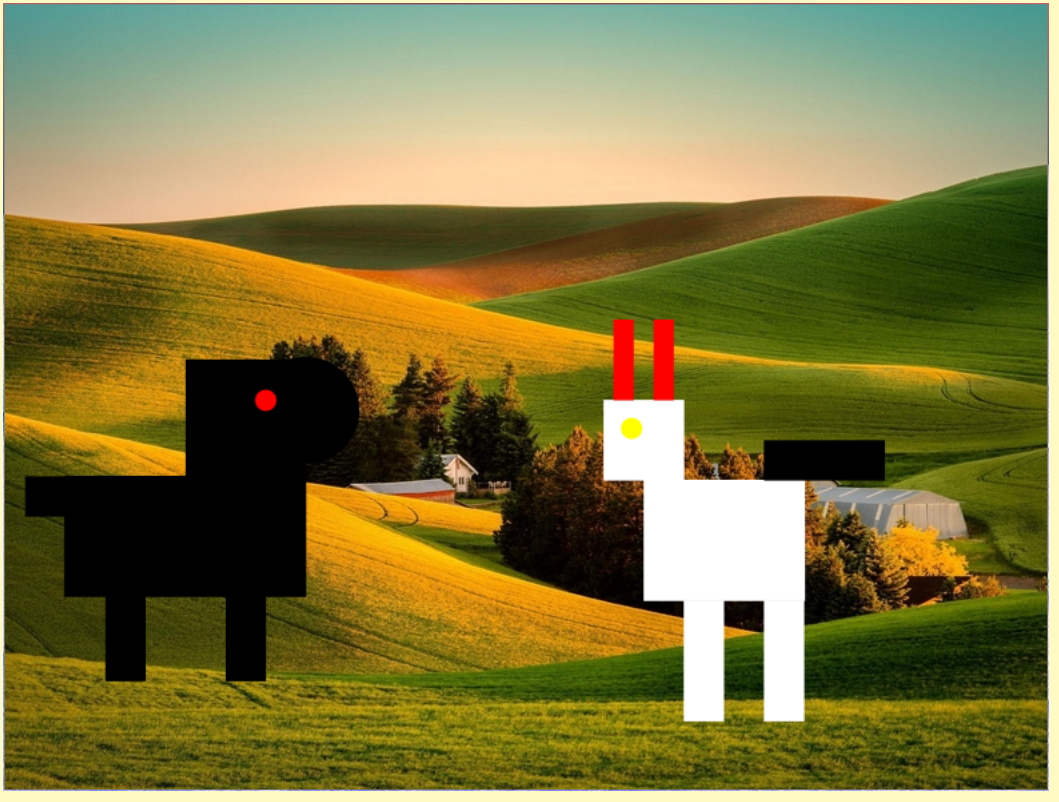


The goat rotates to reach its food faster.

```
if (timeline < sceneOne) {
     tx= tx+timeline;
    cairo_translate(cr, -1, -1);
    cairo_rotate(cr,((-2*M_PI)/180));
    cairo_translate(cr,(0.25*x)+2,-1 *y);
            drawGoat(cr,(0.25*x)+8,(-1 *y)-10);
}
//cairo_scale(cr,sx,sy);
```

When the timeline is in the second scene, the wolf approaches the goat.

The animation continues to Scene 2 where there is a goat who is joined by a wolf on the screen.

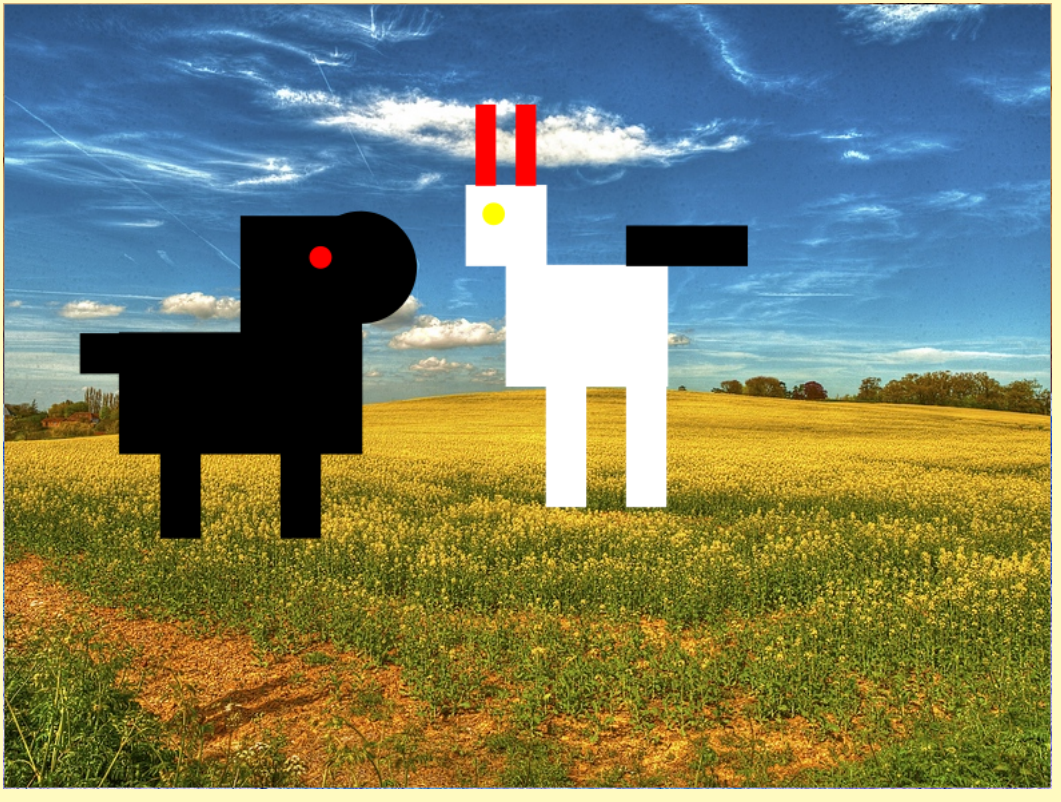


The wolf's mouth is increasing in size as his appetite increases.

```
if (timeline >= sceneOne && timeline <= sceneTwo) {
    tx= tx+timeline;
    //cout << timeline << endl;
    drawWolf(cr,pts2[timeline].x,pts2[timeline].y); //ensures separate
        movement from special data and Bezier curve
    //cout << pts[timeline].x << ":" << pts[timeline].y << endl;
    drawGoat(cr,pts[timeline].x,pts[timeline].y); //ensures separate
        movement from special data and Bezier curve

}
//std::cout<<"tx is now: "<<tx<<std::endl;

cairo_translate(cr,tx,ty);
//  cairo_scale(cr,sx,sy);
```

When the timeline is in the third scene, the wolf and goat dance together.

The animation continues with Scene 3 where the goat and wolf move together as they dance.

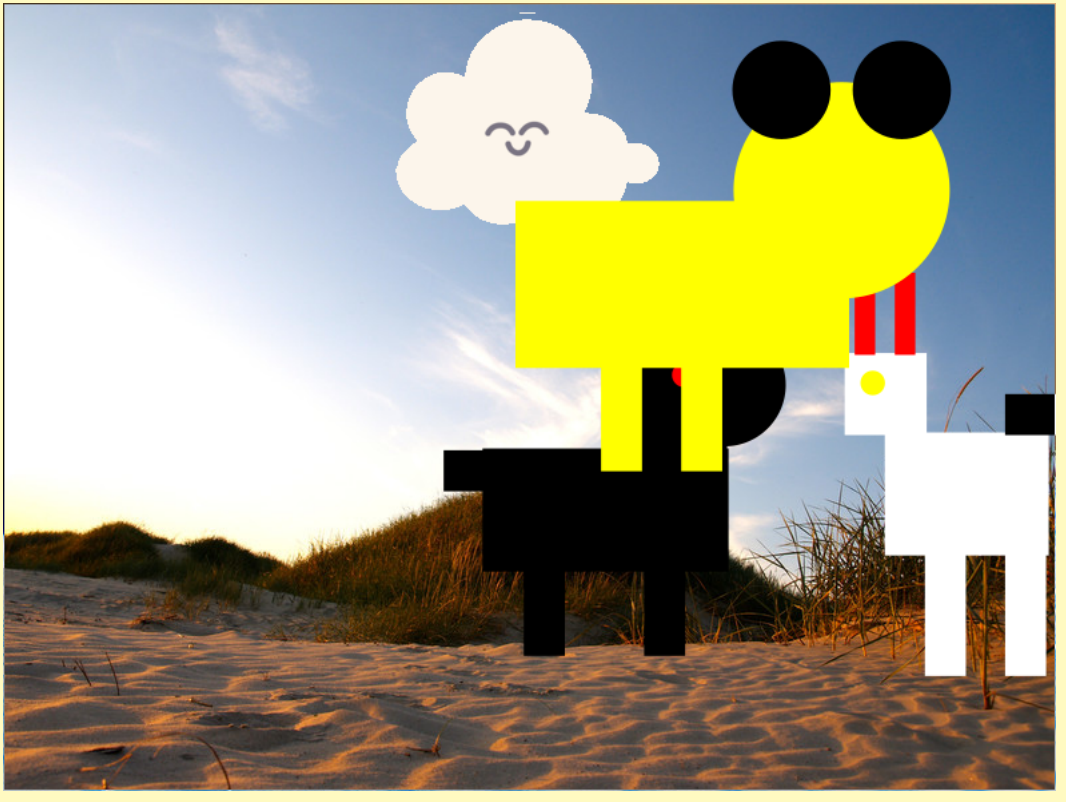


```
if (timeline > sceneTwo && timeline<= sceneThree) {
    if (nopause) tx= tx+timeline;
    cout <<tx<<endl;
```

```
        //cairo_save(cr);
        drawWolf(cr,.25*x,-1.5*y);
        //cw2.redraw();

        //cairo_restore(cr);

       int ptsIndex = ((timeline % 10) * 10); //making the movement more
           crisp

        drawGoat(cr,pts3[ptsIndex].x,pts3[ptsIndex].y); //ensures separate
            movement from special data and Bezier curve
    }
    //cairo_translate(cr,tx,ty);
    //  cairo_scale(cr,sx,sy);
    //std::cout<<"tx is now: "<<tx<<std::endl;
```

The fourth and final scene shows the dog attacking the wolf allowing the goat to escape.

The animation ends with scene 4 where the dog chases the wolf and the goat gets to escape.



```
if (timeline > sceneFour) {
    if (nopause) tx=tx+timeline;
    drawWolf(cr,.25*x,((-1.5*y)-3));
```

```
            drawGoat(cr,.25*x,((-1*y)-3));
            //cairo_save(cr);
            drawDog(cr,0.1*x,0.1*y);
            //cairo_restore(cr);
        }
        cairo_translate(cr,tx,ty);
        // cairo_scale(cr,sx,sy);



}
```

4.4 lab.h

```
#include "config.h"
#include <FL/Fl_Cairo_Window.H>
#include <FL/Fl_Button.H>
#include <iostream>
#include <cmath>
#include <math.h>
#include <string>
#include <fstream>
```

```
   Owner:  Tanmay
```

```
   All information that needs to be repeated is put into file lab.h so
that an include statemnt can be used for the information rather than retyping
it several times in all of the different function files.
```

```
using namespace std;
```

```
   unit is pixels (picture elements
const means "constant" and indicates to
the compiler, that we will not modify
these values during our program exectution
```

```
const int WIDTH = 801;
```

```
const int HEIGHT = 601;
```

```
   timeline is a global variable but extern makes this
only a declaration rather than a definition
so that we won't get multiple definition errors and same for cw
```

```
extern int timeline;
extern bool nopause; // this is a declaration
extern Fl_Cairo_Window cw1;
extern Fl_Cairo_Window cw2;
void drawCB1(Fl_Cairo_Window* cw,cairo_t* cr);
void drawCB2(Fl_Cairo_Window* cw,cairo_t* cr);
void drawHead(cairo_t* cr,double x, double y);
void drawEyes(cairo_t* cr,double x, double y);
void drawBody(cairo_t* cr,double x, double y);
void drawTail(cairo_t* cr,double x, double y);
void drawLeg1(cairo_t* cr,double x, double y);
void drawLeg2(cairo_t* cr,double x, double y);
void drawWolf(cairo_t* cr,double x, double y);
void gHead(cairo_t* cr,double x, double y);
void gBody(cairo_t* cr,double x, double y);
void drawGoat(cairo_t* cr,double x, double y);
void gEye(cairo_t* cr,double x, double y);
void gLeg1(cairo_t* cr,double x, double y);
void gLeg2(cairo_t* cr,double x, double y);
void gTail(cairo_t* cr,double x, double y);
void gHorn1(cairo_t* cr,double x, double y);
void gHorn2(cairo_t* cr,double x, double y);
void drawNose(cairo_t* cr,double x, double y);
void drawDog(cairo_t* cr,double x, double y);
void drawdbody(cairo_t* cr,double x, double y);
```

```
void drawdhead(cairo_t* cr,double x, double y);
void drawdear1(cairo_t* cr,double x, double y);
void drawdear2(cairo_t* cr,double x, double y);
void drawdleg1(cairo_t* cr,double x, double y);
void drawdleg2(cairo_t* cr,double x, double y);
void soundEffects();
double mirrorY(double y);
double scaler(double u);
void moveCharacter(void*);
void drawBackground(cairo_t* cr);
void drawBirds(cairo_t* cr);
void pauseCB(Fl_Widget *, void*);
void showText(cairo_t* cr, std::string s);
string readFable();

void drawBackground1(cairo_t* cr);
void drawBackground2(cairo_t* cr);
void drawBackground3(cairo_t* cr);
void drawCloud(cairo_t* cr);



const int sceneOne = 10;
const int sceneTwo = 20;
const int sceneThree = 25;
const int sceneFour = 30;
//structure created for the points from gnuplot
struct Points{
        double x,y;
};
void getPoints();
```

```
extern Points pts[]; //points from gnuplot
void getPoints2();
extern Points pts2[];
void getPoints3();
extern Points pts3[];


void getBirds();
extern std::string s[];
const int N = 6;
```

4.5 drawhead.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Anusha
```

```
The head of the wolf is drawn to the graphics window.
```

```
void drawHead(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

```
rgba where alpha is amount of transparency
```

```
        cairo_set_source_rgb (cr, 0,0,0); // black
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+2), mirrorY(scaler(y+8)), ((.0005*
            timeline)+1)*scaler(3),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.6 drawEyes.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Anusha
```

```
The eyes of the wolf are drawn to the graphics window.
```

```
void drawEyes(cairo_t* cr,double x, double y)
{
        //std::cout<<"drawing the eyes"<<std::endl;
        cairo_set_source_rgb(cr,1, 0,0);
```

```
x,y,r,start,end of arc in degrees
```

```
        cairo_arc(cr, scaler(x+4), mirrorY(scaler(y+10)), ((.005*timeline)
            +1)*scaler(0.25), 0,2*M_PI);
        cairo_fill(cr);
}
```

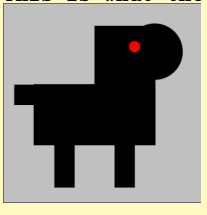
4.7 drawWolf.cpp

```
#include "lab.h"
```

```
Owner:  Anusha
```

```
   This connects the many drawings of the wolf
to make the wolf.
```

This is what the wolf looks like:



```
void drawWolf(cairo_t* cr,double x, double y)
{
     drawHead(cr,x,y);
     drawBody(cr,x,y);
     drawTail(cr,x,y);
     drawLeg1(cr,x,y);
     drawLeg2(cr,x,y);
```

```
        drawNose(cr,x,y);
        drawEyes(cr,x,y);
    }
```

4.8 mirrorY.cpp

```
#include "lab.h"
```

```
Owner:  Sarah
```

```
Mirror image of y coordinate because the
cartesian coordinate system typically has
origin at lower left corner but in the PC
screen, the origin is in the upper left corner.
```

```
double mirrorY(double y)
{
    //std::cout<<"in mirrorY y is: " <<y<<std::endl;
    return HEIGHT - y;
}
```

4.9 scaler.cpp

```
#include "lab.h"
```

```
Owner:  Phone
```

```
scale the unit size used for x y coordinates
example:  if u is 2, the function returns
HEIGHT/10*2 = 40 (pixels)
for HEIGHT = 200
```

```
double scaler(double u)
{
    //std::cout << "in scaler u is: " <<u <<std::endl;
    return HEIGHT / 10.0 * u;
}
```

4.10 moveCharacter.cpp

```
#include "lab.h"
```

```
Owner:  Tanmay
```

```
This is the translate part of the animation and will cause the characters to move.
```

```
void moveCharacter(void*)
{
    //cw2.show();
    //std::cout<<"time is now: "<<timeline<<std::endl;
    if (nopause)
    {
        timeline= timeline+1;
    }

    soundEffects();
```

```
If the timeline reaches the time assigned, then the story would change the
subtitles to match the scene that is playing.
```

Add one to scene timings to ensure that everything redraws correctly

```
    if (timeline == (sceneOne + 1)) {
        cw1.redraw();
    } else if (timeline == (sceneTwo + 1)) {
        cw1.redraw();
    } else if (timeline == (sceneThree + 1)) {
        cw1.redraw();
    } else if (timeline == (sceneFour + 1)) {
        cw1.redraw();
    } else {
        cw2.redraw();
    }
```

repeat this function every 1 second

```
    Fl::repeat_timeout(1, moveCharacter);
}
```

4.11 drawBackground.cpp

```
#include "lab.h"
```

Owner: Phone

This gives the second window a background of a pasture for the first scene so that the characters
and birds are moving on top of the background rather than a blank screen.

This is what the background looks like:

```
void drawBackground(cairo_t* cr)
{
    int w, h;
    cairo_surface_t* image;
    //std::cout << "moving"<<std::endl;
    image = cairo_image_surface_create_from_png ("field.png");
    w= cairo_image_surface_get_width (image);
    h=cairo_image_surface_get_height (image);
```

```
    //cairo_scale (cr, 800.0/w, 600.0/h);
    cairo_set_source_surface (cr, image, 0,0);
    cairo_paint (cr);
    cairo_surface_destroy (image);
}
```

4.12 drawBackground1.cpp

```
#include "lab.h"
```

```
Owner:  Phone
```

```
   This gives the second window a background of the second scene so that the characters
are moving on top of the background rather than a blank screen.
```

```
This is what the background looks like:
```

```
void drawBackground1(cairo_t* cr)
{
    int w, h;
    cairo_surface_t* image;
    image = cairo_image_surface_create_from_png ("field2.png");
    w= cairo_image_surface_get_width (image);
    h=cairo_image_surface_get_height (image);
    cairo_set_source_surface (cr, image, 0,0);
    cairo_paint (cr);
    cairo_surface_destroy (image);
}
```

4.13 drawBackground2.cpp

```
#include "lab.h"
```

```
Owner:  Phone
```

```
   This gives the second window a background of the third scene so that the characters
are moving on top of the background rather than a blank screen.
```

```
This is what the background looks like:
```

```
void drawBackground2(cairo_t* cr)
{
    int w, h;
    cairo_surface_t* image;
    //std::cout << "moving"<<std::endl;
    image = cairo_image_surface_create_from_png ("field3.png");
    w= cairo_image_surface_get_width (image);
    h=cairo_image_surface_get_height (image);
    //cairo_scale (cr, 800.0/w, 600.0/h);
    cairo_set_source_surface (cr, image, 0,0);
    cairo_paint (cr);
    cairo_surface_destroy (image);
}
```

4.14 drawBackground3.cpp

```
#include "lab.h"
```

Owner: Phone

This gives the second window a background of the last scene so that the characters and clouds are moving on top of the background rather than a blank screen.

This is what the background looks like:

```
void drawBackground3(cairo_t* cr)
{
    int w, h;
    cairo_surface_t* image;
    //std::cout << "moving"<<std::endl;
    image = cairo_image_surface_create_from_png ("field4.png");
    w= cairo_image_surface_get_width (image);
    h=cairo_image_surface_get_height (image);
    //cairo_scale (cr, 800.0/w, 600.0/h);
```

```
    cairo_set_source_surface (cr, image, 0,0);
    cairo_paint (cr);
    cairo_surface_destroy (image);
}
```

4.15 drawBirds.cpp

```
#include "lab.h"
```

```
Owner:  Anusha
```

```
This is the animation of the birds moving around at the top of the window.
```

This is what the birds look like:

```
const int N = 6; //vary depending on particular animated gif
std::string s[N] =
"birds/birdsn.png",
"birds/birdsn1.png",
"birds/birdsn2.png",
"birds/birdsn3.png",
```

```
"birds/birdsn4.png",
"birds/birdsn5.png"
;
```

```
void drawBirds(cairo_t* cr)
{
    int w, h;
    cairo_surface_t* image;
    //std::cout << "moving"<<std::endl;
    std::string imageFile = "birds/" + s[timeline%N];
    image = cairo_image_surface_create_from_png (imageFile.c_str());
    w= cairo_image_surface_get_width (image);
    h=cairo_image_surface_get_height (image);
    //cairo_scale (cr, 100.0/w, 100.0/h);
    cairo_set_source_surface (cr, image, WIDTH/2-w/2,0);
    cairo_paint (cr);
    cairo_surface_destroy (image);
}
```

4.16 getBirds.cpp

```
#include "lab.h"
#include <fstream>
```

```
   Owner:  Tanmay
```

```
   This allows us to see the GIF of the birds run
in the first scene through an if statement to see
if there is outupt in the array to run the animation.
```

```
//int N = 6; //vary depending on particular animated gif
std::string s[N];

void getBirds() {

    system("ls birds > birdsOut");
    ifstream file("birdsOut");

    std::string empty;
    file >> empty;
```

```
   only enter the if statement if reading the argument
and then carry on to the for loop leading to an array
```

```
        if (file.is_open()) {

            for (int i = 0; i < N; i++) {
                file >> s[i];
            }
        }
```

```
std::string s[N] =
"birds/birdsn.png",
"birds/birdsn1.png",
"birds/birdsn2.png",
"birds/birdsn3.png",
"birds/birdsn4.png",
"birds/birdsn5.png"
;
```

```
    }
```

4.17 drawCloud.cpp

```
#include "lab.h"
```

```
Owner:  Anusha
```

```
This is the animation of the cloud moving around at the top of the window.
```

```
This is what the cloud looks like:
```

```
const int C = 8; //vary depending on particular animated gif
std::string q[C] = {

    "cloud/cloud07.png",
    "cloud/cloud15.png",
    "cloud/cloud23.png",
    "cloud/cloud31.png",
    "cloud/cloud39.png",
    "cloud/cloud47.png",
    "cloud/cloud63.png",
```

```
    "cloud/cloud71.png"

};



void drawCloud(cairo_t* cr)
{
    int w, h;
    cairo_surface_t* image;
    image = cairo_image_surface_create_from_png (q[timeline%C].c_str());
    w= cairo_image_surface_get_width (image);
    h=cairo_image_surface_get_height (image);
    cairo_set_source_surface (cr, image, WIDTH/2-w/2,0);
    cairo_paint (cr);
    cairo_surface_destroy (image);
}
```

4.18 drawBody.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Anusha
```

```
The head of the wolf is drawn to the graphics window.
```

```
void drawBody(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

```
rgba where alpha is amount of transparency
```

```
        cairo_set_source_rgb (cr, 0,0,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x-1.03), mirrorY(scaler(y+5.1)),
            ((.0005*timeline)+1)*scaler(6),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.19 drawdbody.cpp

Owner: Phone

The body of the dog is drawn to the graphics window.

```
#include "lab.h"
#include <iostream>
void drawdbody(cairo_t* cr,double x, double y)
{
    cairo_set_source_rgb (cr, 1, 1, 0);
    cairo_rectangle(cr, -5, 10, 500, 250);
    cairo_fill (cr);
}
```

4.20 drawdear1.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Phone
```

```
The left ear of the dog is drawn to the graphics window.
```

```
void drawdear1(cairo_t* cr,double x, double y)
{
        //std::cout<<"drawing the eyes"<<std::endl;
        cairo_set_source_rgb(cr,0, 0,0);
```

```
x,y,r,start,end of arc in degrees
```

```
        cairo_arc(cr, scaler(x+6.5), mirrorY(scaler(y+12.5)), ((.005*
            timeline)+1)*scaler(1), 0,2*M_PI);
        cairo_fill(cr);
}
```

4.21 drawdear2.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Phone
```

```
The right ear of the dog is drawn to the graphics window.
```

```
void drawdear2(cairo_t* cr,double x, double y)
{
        //std::cout<<"drawing the eyes"<<std::endl;
        cairo_set_source_rgb(cr,0, 0,0);
```

```
x,y,r,start,end of arc in degrees
```

```
        cairo_arc(cr, scaler(x+9.5), mirrorY(scaler(y+12.5)), ((.005*
            timeline)+1)*scaler(1), 0,2*M_PI);
        cairo_fill(cr);
}
```

4.22 drawdhead.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Phone
```

```
The dog's head is drawn to the graphics window.
```

```
void drawdhead(cairo_t* cr,double x, double y)
{
        //std::cout<<"drawing the eyes"<<std::endl;
        cairo_set_source_rgb(cr,1, 1,0);
```

```
x,y,r,start,end of arc in degrees
```

```
        cairo_arc(cr, scaler(x+8), mirrorY(scaler(y+10)), ((.005*timeline)
            +1)*scaler(2.2), 0,2*M_PI);
        cairo_fill(cr);
}
```

4.23 drawdleg1.cpp

```
#include "lab.h"
#include <cmath>
```

Owner: Phone

The dog's left leg is drawn to the graphics window.

```
void drawdleg1(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

rgba where alpha is amount of transparency

```
        cairo_set_source_rgb (cr, 1,1,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+2), mirrorY(scaler(y+3)), ((.0005*
            timeline)+1)*scaler(1),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.24 drawdleg2.cpp

```
#include "lab.h"
#include <cmath>
```

Owner: Phone

The dog's right leg is drawn to the graphics window.

```
void drawdleg2(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

rgba where alpha is amount of transparency

```
        cairo_set_source_rgb (cr, 1,1,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+4), mirrorY(scaler(y+3)), ((.0005*
            timeline)+1)*scaler(1),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

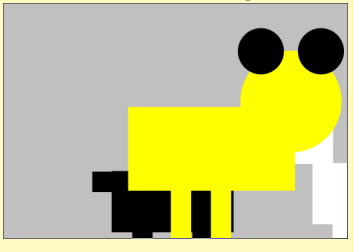
4.25 drawDog.cpp

```
#include "lab.h"
#include <iostream>
```

Owner: Phone

All of the functions to draw the dog are combined in this function to draw the total dog.

This is what the dog looks like:



```
void drawDog(cairo_t* cr,double x, double y)
{
        drawdbody(cr,x,y);
        drawdhead(cr,x,y);
        drawdear1(cr,x,y);
```

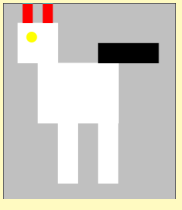
```
        drawdear2(cr,x,y);
        drawdleg1(cr,x,y);
        drawdleg2(cr,x,y);
}
```

4.26 drawGoat.cpp

Owner: Sarah

All of the functions to draw the goat are combined in this function to draw the total goat

This is what the goat looks like:



```
#include "lab.h"
void drawGoat(cairo_t* cr,double x, double y)
{
        gBody(cr,x,y);
        gHead(cr,x,y);
        gEye(cr,x,y);
        gLeg1(cr,x,y);
        gLeg2(cr,x,y);
```

```
        gTail(cr,x,y);
        gHorn1(cr,x,y);
        gHorn2(cr,x,y);
}
```

4.27 drawLeg1.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Anusha
```

```
The wolf's left leg is drawn to the graphics window.
```

```
void drawLeg1(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

```
rgba where alpha is amount of transparency
```

```
        cairo_set_source_rgb (cr, 0,0,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+3), mirrorY(scaler(y+3)), ((.0005*
            timeline)+1)*scaler(1),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.28 drawLeg2.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Anusha
```

```
The wolf's right leg is drawn to the graphics window.
```

```
void drawLeg2(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

```
rgba where alpha is amount of transparency
```

```
        cairo_set_source_rgb (cr, 0,0,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x), mirrorY(scaler(y+3)), ((.0005*
            timeline)+1)*scaler(1),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.29 drawNose.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Anusha
```

```
The wolf's nose is drawn to the graphics window.
```

```
void drawNose(cairo_t* cr,double x, double y)
{
        //std::cout<<"drawing the eyes"<<std::endl;
        cairo_set_source_rgb(cr,0, 0,0);
```

```
x,y,r,start,end of arc in degrees
```

```
        cairo_arc(cr, scaler(x+5), mirrorY(scaler(y+9.75)), ((.005*timeline)
            +1)*scaler(1.25), 0,180);
        cairo_fill(cr);
}
```

4.30 drawTail.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Anusha
```

```
The wolf's tail is drawn to the graphics window.
```

```
void drawTail(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

```
rgba where alpha is amount of transparency
```

```
        cairo_set_source_rgb (cr, 0,0,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x-2), mirrorY(scaler(y+7.1)), ((.0005*
            timeline)+1)*scaler(3),((.0005*timeline)+1)*scaler(-1));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.31 gBody.cpp

```
#include "lab.h"
#include <cmath>
```

Owner: Sarah

The goat's body is drawn to the graphics window.

```
void gBody(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

rgba where alpha is amount of transparency

```
        cairo_set_source_rgb (cr, 1,1,1); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+9), mirrorY(scaler(y+5)), ((.0005*
            timeline)+1)*scaler(4),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.32 gEye.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Sarah
```

```
The goat's eye is drawn to the graphics window.
```

```
void gEye(cairo_t* cr,double x, double y)
{
        //std::cout<<"drawing the eyes"<<std::endl;
        cairo_set_source_rgb(cr,1, 1,0);
```

```
x,y,r,start,end of arc in degrees
```

```
        cairo_arc(cr, scaler(x+8.7), mirrorY(scaler(y+9.3)), ((.005*timeline
            )+1)*scaler(0.25), 0,2*M_PI);
        cairo_fill(cr);
}
```

4.33 gHead.cpp

```
#include "lab.h"
#include <cmath>
```

Owner: Sarah

The head of the goat is drawn to the graphics window.

```
void gHead(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

rgba where alpha is amount of transparency

```
        cairo_set_source_rgb (cr, 1,1,1); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+8), mirrorY(scaler(y+8)), ((.0005*
            timeline)+1)*scaler(2),((.0005*timeline)+1)*scaler(-2));
        //cairo_rectangle (cr, 8,8, 2,2);
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.34 gHorn1.cpp

```
#include "lab.h"
#include <cmath>
```

Owner: Sarah

The goat's left horn is drawn to the graphics window.

```
void gHorn1(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

rgba where alpha is amount of transparency

```
        cairo_set_source_rgb (cr, 1,0,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+9.25), mirrorY(scaler(y+10)), ((.0005*
            timeline)+1)*scaler(0.5),((.0005*timeline)+1)*scaler(-2));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.35 gHorn2.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Sarah
```

```
The goat's right horn is drawn to the graphics window.
```

```
void gHorn2(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

```
rgba where alpha is amount of transparency
```

```
        cairo_set_source_rgb (cr, 1,0,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+8.25), mirrorY(scaler(y+10)), ((.0005*
            timeline)+1)*scaler(0.5),((.0005*timeline)+1)*scaler(-2));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.36 gLeg1.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Sarah
```

```
The goat's left leg is drawn to the graphics window.
```

```
void gLeg1(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

```
rgba where alpha is amount of transparency
```

```
        cairo_set_source_rgb (cr, 1,1,1); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+10), mirrorY(scaler(y+2)), ((.0005*
            timeline)+1)*scaler(1),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.37 gLeg2.cpp

```
#include "lab.h"
#include <cmath>
```

Owner: Sarah

The goat's right horn is drawn to the graphics window.

```
void gLeg2(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

rgba where alpha is amount of transparency

```
        cairo_set_source_rgb (cr, 1,1,1); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+12), mirrorY(scaler(y+2)), ((.0005*
            timeline)+1)*scaler(1),((.0005*timeline)+1)*scaler(-3));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.38 gTail.cpp

```
#include "lab.h"
#include <cmath>
```

```
Owner:  Sarah
```

```
The tail of the goat is drawn to the graphics window.
```

```
void gTail(cairo_t* cr,double x, double y)
{
        //std::cout<< "drawing the head" << std::endl;
```

```
rgba where alpha is amount of transparency
```

```
        cairo_set_source_rgb (cr, 0,0,0); // maroon
        //std::cout<<"x is: " <<x<<" "<<"y is: "<<y<<std::endl;
        //std::cout<<"scaled x is: " <<scaler(x) <<" ";
        //std::cout<<"scaled y is: "<<mirrorY(scaler(y))<<std::endl;
        cairo_rectangle (cr, scaler(x+12), mirrorY(scaler(y+8)), ((.0005*
            timeline)+1)*scaler(3),((.0005*timeline)+1)*scaler(-1));
```

```
   Since positive y goes down from upper part of screen, then height
is negated to make it go up.

            cairo_fill (cr);
   }
```

4.39 fables.txt

```
There was once a little Kid who thought he
 was strong because of his growing horns.
 While the Kid was eating , his flock disappears
 and a wolf appears causing the Kid to bargain
 for his life. The Kid performs for the wolf as the
 wolf pipes a tune and they dance. The Shepherds
 Dog hears the piping and jumps on top of the Wolf to
 attack him; the moral is to not let anything turn you
 from your purpose.
```

4.40 readFable.cpp

```
   Owner:  Tanmay
```

```
   #include <iostream>
   #include <fstream>
```

```
   Our fable is read from a file rather than written
directly on the screen.
```

```
   using namespace std;

   string readFable()
   {

        // Read data from fables.txt
        ifstream file("fables.txt");
```

```
   Variable setup

str - holds each line of file while looping

file_contents - contents of entire "fables.txt"
```

```
string str;
string file_contents;

// Loop until there are no more lines in the file
while (getline(file, str))
{
    // Add each line of file to file\_contents to create string with
        entire file
    file_contents += str;
}

//int arrSize = (file_contents.size()+1000);
//cout << "arrSize: " << arrSize << endl;

string fable;

// Create a boolean which we will use later to store whether or not the
    phrase has any vowels
bool containsVowel;
// Loop over every character in file\_contents
for (int i = 0; i < file_contents.size(); i++)
{
    // Check each character for punctuation
    if (file_contents[i] == '.' || file_contents[i] == '!' ||
        file_contents[i] == '?')
    {
        char c;
        int start = i;
        int stop;
        // char c - used to loop over sequence of letters separated by
            spaces (a word or abbreviation)
        // int start - holds position of space character which precedes
```

```
    the word
// int stop - holds position of punctuation, which is the
    current index

while (true)
{
    if (file_contents[start] == ' ')
    {
        break;
    }
    start--;
}

// Set stop variable to start position so we start the loop at
    the beginning of the word/abbreviation
stop = start;
// Loop forwards from the start position until the stop position
    , which again is the current index
while (stop != i)
{
    // c holds a single character, which at the beginning of the
        loop is the first letter of the word/abbreviation
    // As the loop goes on, c is the second and third and fourth
        letter and so on until the word/abbreviation ends
    c = file_contents[stop];
    // Check if c is a vowel
    if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u'
        || c == 'A' || c == 'E' || c == 'I' || c == 'O' || c ==
        'U')
    {
        // If c is a vowel at any time in the loop, then the
            word has a vowel in it
```

```
                // Set boolean containsVowel to true so we can check for
                    it later
                containsVowel = true;
            }
            // Increment the stop variable so c becomes the next
                character in the word/abbreviation
            stop++;
        }

        // Check if the loop found any vowels in the word, and if so,
            print a newline after it
        // The logic here is that in English, all words have vowels but
            abbreviations do not follow that rule
        if (containsVowel)
        {
            // Add a new line for lines with punctuation
            fable += file_contents[i];
            fable += '\n';
            // Because punctuation is generally followed by a space,
                skip the next character in the list
            i++;
            //fable[i] = '\n';
        } else {
            // If the loop did not find any vowels and did not set
                containsVowel to false, then the punctuation is most
                likely part an abbreviation and is not the end of a line
            // Print the punctuation with no newline
            fable += file_contents[i];
        }
    }
    else
    {
```

```
            // If there is no punctuation, then just display the character
            fable += file_contents[i];
        }
        containsVowel = false;
    }
    // Terminate program


    return fable;
}
```

4.41 showText.cpp

```
   Owner:  Tanmay
```

```
   The text of our fable is shown on a screen that is
right next to that of the animation and which is time synced.
```

```
   #include "lab.h"

   void showText(cairo_t* cr, std::string s)
   {
```

```
   Create extents object
```

```
       cairo_text_extents_t extents;
```

```
   Start at beginning of screen
```

```
       int xPos = 0;
```

```
Start at 40 due to scaling issues
```

```
    int yChange = 40;
```

```
Create empty string variable for the current line
```

```
    std::string line = "";
```

```
Move the first line object to the start position
```

```
    cairo_move_to(cr, xPos, yChange);
```

```
Loop over each character in s
```

```
    int i = 0;
    while (i < s.size())
    {
```

Start another loop inside the first one which will put together the lines and make sure they are not too big to display on the screen

```
        while (true)
        {
```

Store the location of the start of the word for later

```
            int startOfWord = i;
```

Start -another- loop which finds the end of each word in the line

Loop forever until.....

```
            while (true)
            {
                //std::cout << "i: " << i << std::endl;
                //std::cout << "s[i]" << s[i] << std::endl;
```

Increment the index

```
        i++;
```

Until we reach a space, indicating the end of the word

```
        if (s[i] == ' ') break;
    }
    //std::cout << "Start of word position: " << startOfWord << std
        ::endl;
    std::string word = "";
```

Loop forwards now from the start of the word, until the end of the word which is our current index i

```
    for (int j = startOfWord; j < i; j++)
    {
        word += s[j];
    }
    //std::cout << "Word: " << word << std::endl;
```

Store and compare the extents.width of both the line and the word

If the word width + the line width exceeds the defined window width in lab.h, then don't add the word to the line, start a new line

```
cairo_text_extents(cr,word.c_str(), &extents);
int wordWidth = extents.width;
cairo_text_extents(cr,line.c_str(), &extents);
int lineWidth = extents.width;
int combined = wordWidth + lineWidth;

//std::cout << "wordWidth: " << wordWidth << " lineWidth: " <<
    lineWidth << " combined: " << combined << std::endl;
```

If the line width + the current word's width is less than "WIDTH" defined in lab.h, then add the word to the current line and show it on the screen

```
if (combined < WIDTH)
{

    line += " " + word;
    //std::cout << "Line: " << line << std::endl;
    cairo_show_text(cr, word.c_str());
    break;
```

If not, and adding the current word to the line would exceed WIDTH, then create a new line and make that word the first of the line

```
                }
                else
                {
                    yChange += extents.height;
                    cairo_move_to(cr, xPos-20, yChange);
                    cairo_show_text(cr, word.c_str());
                    line = word;
                    break;

                }
            }
        }
    }
```

4.42 getPoints.cpp

```
#include "lab.h"
```

```
Owner:  Anusha
```

```
   This function gets points from the data files
that was written in gnuplot and the points
would follow the bezier curve and so it is called
here in order for each character to have its
own path.
```

```
Points pts[100] ; //100 points in point file
void getPoints()
{
        string s;
        ifstream ifs("points"); //data went into points
        for(int i = 0; i <4 ; i++)
        {
                getline(ifs,s);
        }

        double x,y; char c; int i = 0;
        while(ifs >> x >> y >> c)
        {
```

```
                    pts[i].x = x; pts[i].y=y;
                    i++;
            }
    }
```

4.43 getPoints2.cpp

```
#include "lab.h"
```

```
Owner:  Anusha
```

```
   This function gets points2 from the data2 files
that was written in gnuplot and the points2
would follow the bezier curve and so it is called
here in order for each character to have its
own path.
```

```
Points pts2[100] ; //100 points in point file
void getPoints2()
{
        string s;
        ifstream ifs("points2"); //data2 went to points2
        for(int i = 0; i <4 ; i++)
        {
                getline(ifs,s);
        }

        double x,y; char c; int i = 0;
        while(ifs >> x >> y >> c)
        {
                pts2[i].x = x; pts2[i].y=y;
                i++;
```

```
            }
        }
```

4.44 getPoints3.cpp

```
#include "lab.h"
```

```
Owner:  Anusha
```

```
   This function gets points3 from the data3 files
that was written in gnuplot and the points3
would follow the bezier curve and so it is called
here in order for each character to have its
own path.
```

```
Points pts3[100] ; //100 points in point file
void getPoints3()
{
        string s;
        ifstream ifs("points3"); //data3 went to points3
        for(int i = 0; i <4 ; i++)
        {
                getline(ifs,s);
        }

        double x,y; char c; int i = 0;
        while(ifs >> x >> y >> c)
        {
                pts3[i].x = x; pts3[i].y=y;
                i++;
```

```
            }
        }
```

4.45 soundEffects.cpp

```
#include "lab.h"
```

```
Owner:  Tanmay
```

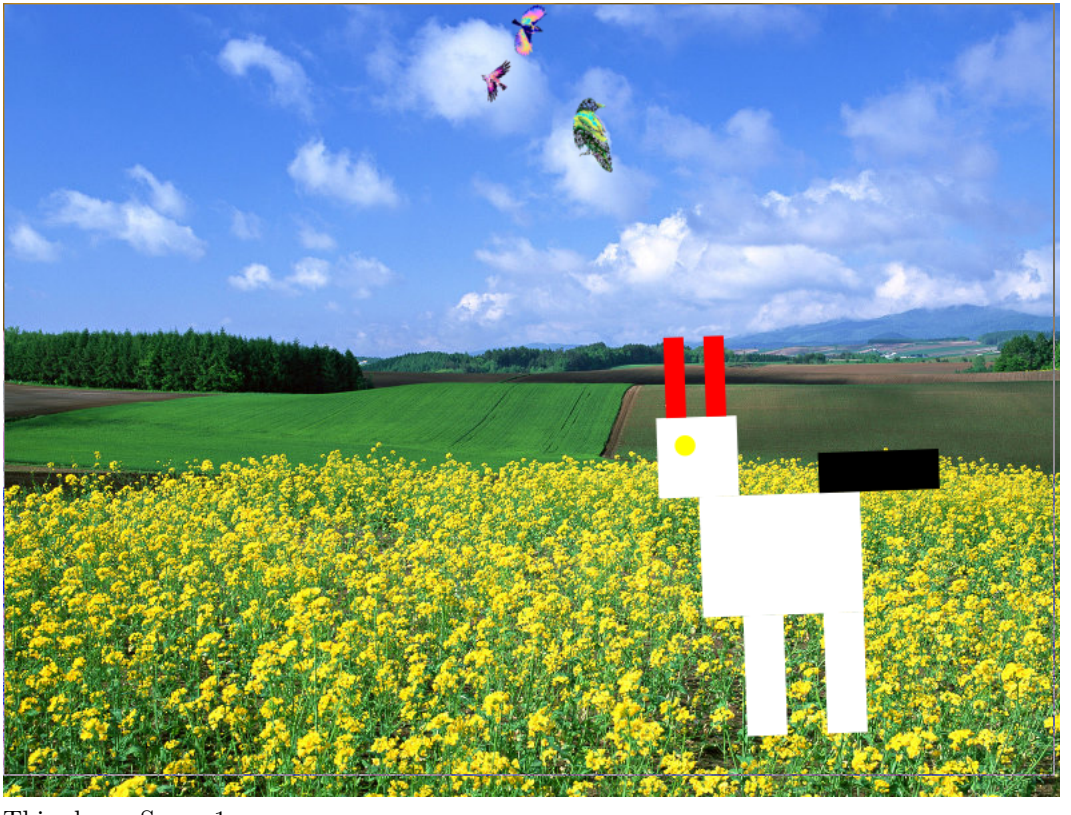
```
   This allows the animation to have
sound to enhance what the characters are
doing.  The sounds play along with the
timeline and approximately when the
scenes change to represent what is going on.
The sound bits were uploaded to our
directory in order for it to work.
```

```
void soundEffects() {

    if (timeline == 1) {
        system("aplay chirping.wav&");
    }

    if (timeline == (sceneTwo-3)) {
        system("aplay grass.wav&");
    }

    if (timeline == (sceneTwo)) {
        system("aplay grass.wav&");
    }
```

```
    if (timeline == sceneThree) {
        system("aplay pipe.wav&");
    }

    if (timeline == (sceneFour-2)) {
        system("aplay growl.wav&");
    }
    if (timeline == (sceneFour)) {
        system("aplay growl.wav&");
    }

}
```

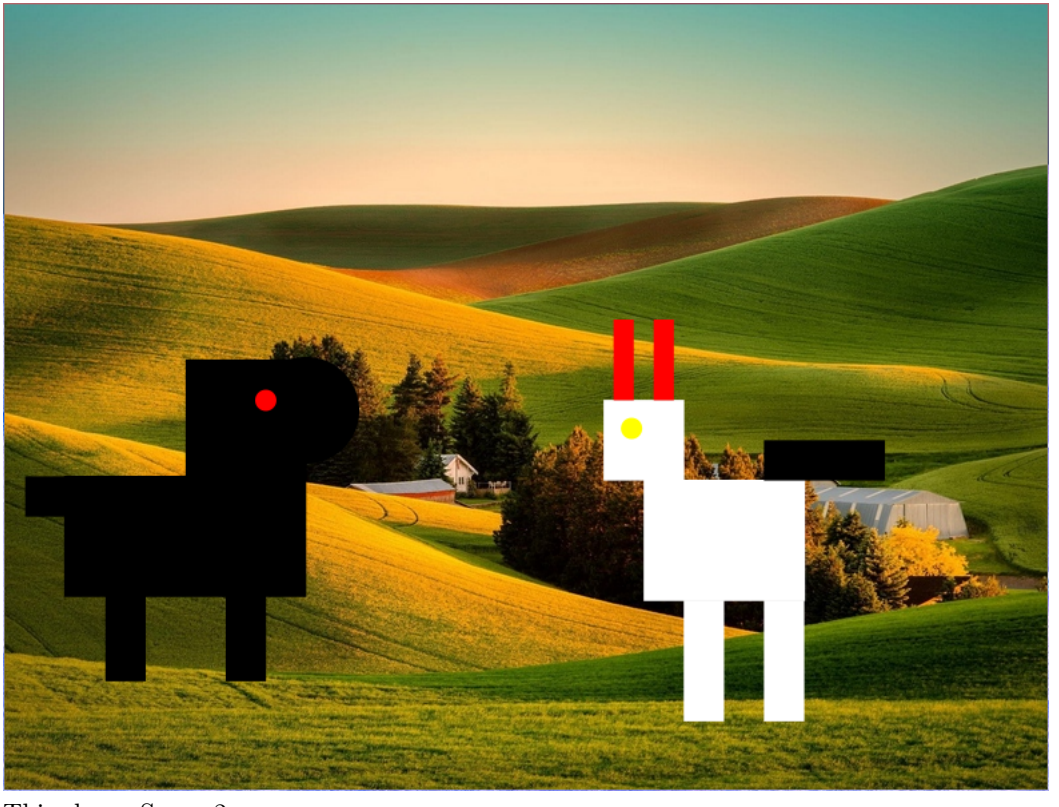
5 Test

5.1 Testcase 1



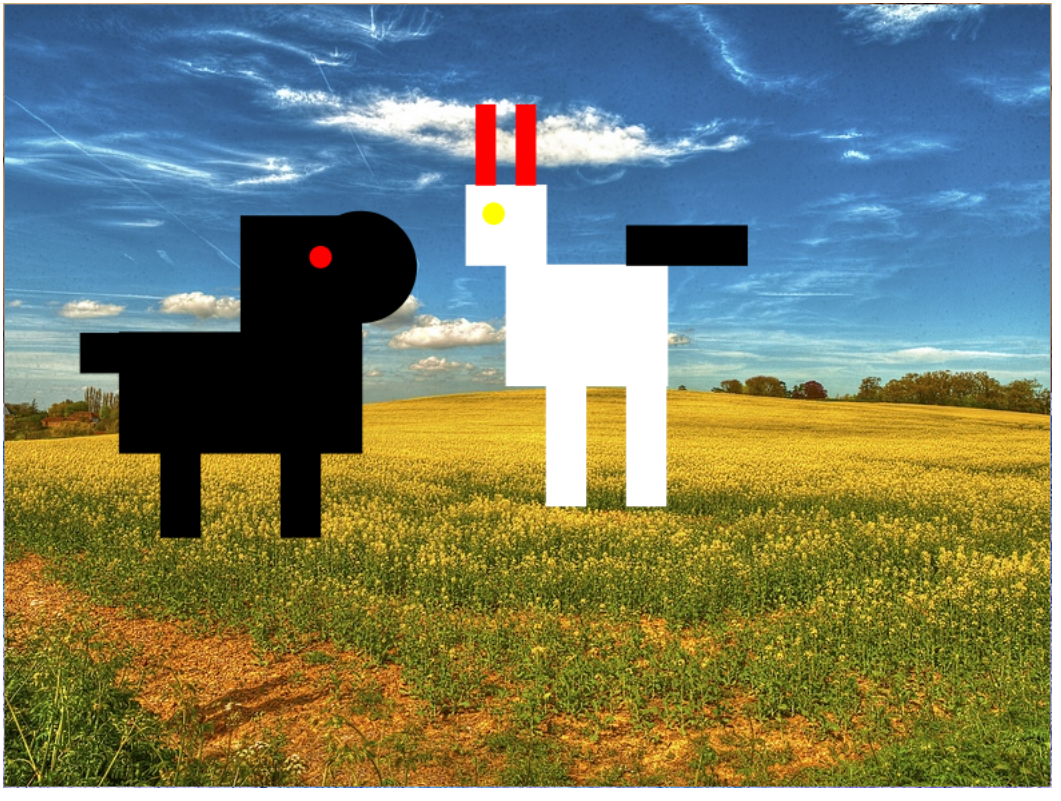
This shows Scene 1.

5.2 Testcase 2



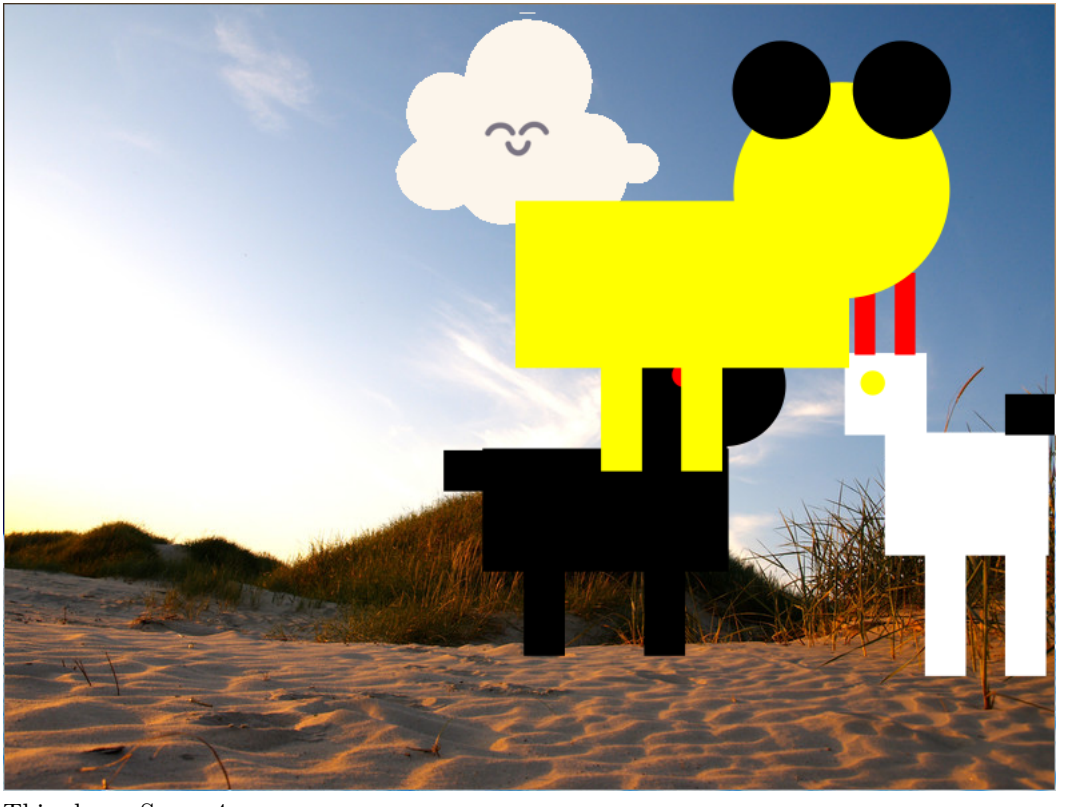
This shows Scene 2.

5.3 Testcase 3



This shows Scene 3.

5.4 Testcase 4



This shows Scene 4.

5.5 Test cont.

The program meets the requreiments with the expected output and the images taken of program running showing the actual results.