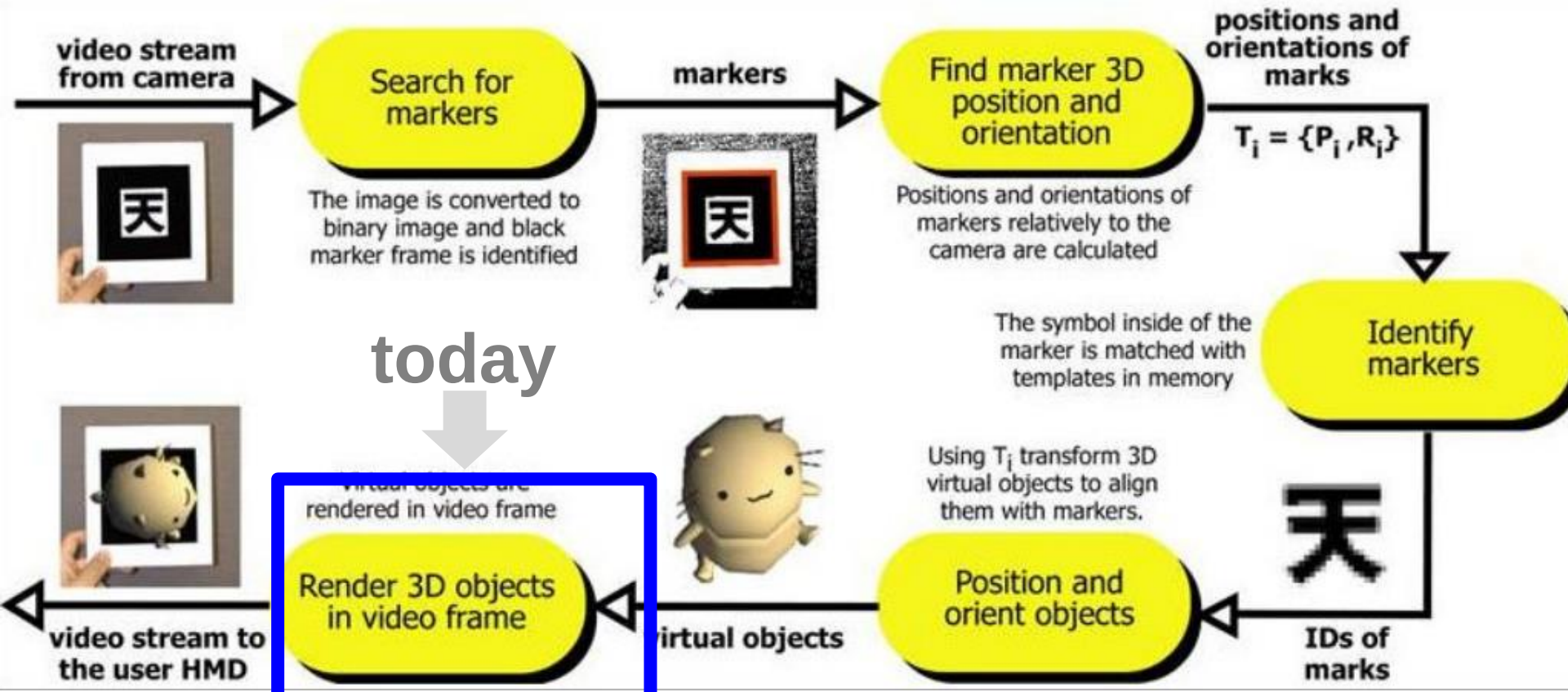
Introduction to Augmented Reality

Tutorial 8: Augmented Reality
June 6, 2018

Andreas Langbein, Adnane Jadid, David Plecher

Marker-based Tracking

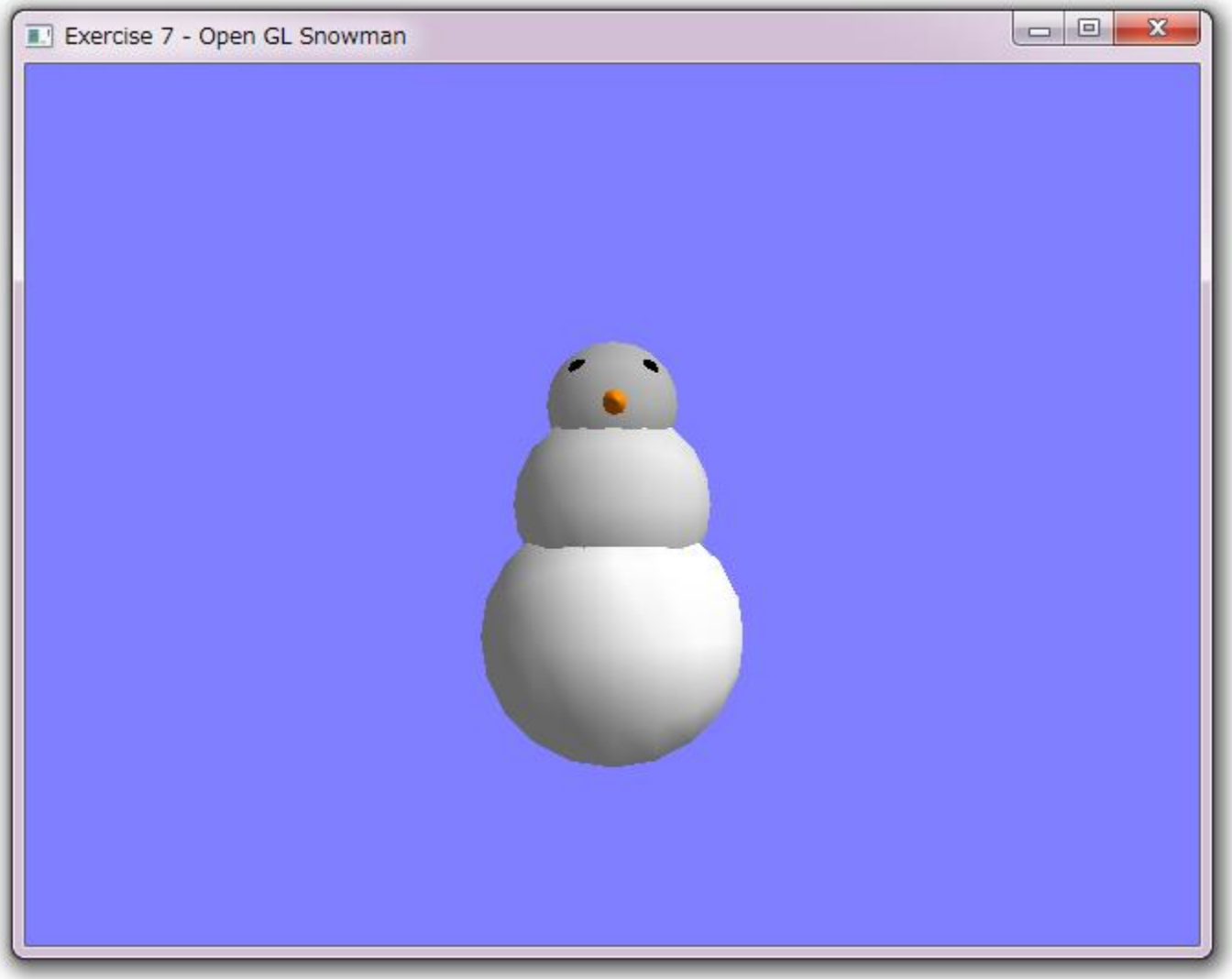


ARToolKit

Ex. 8~9

Solution for the Previous Tutorial

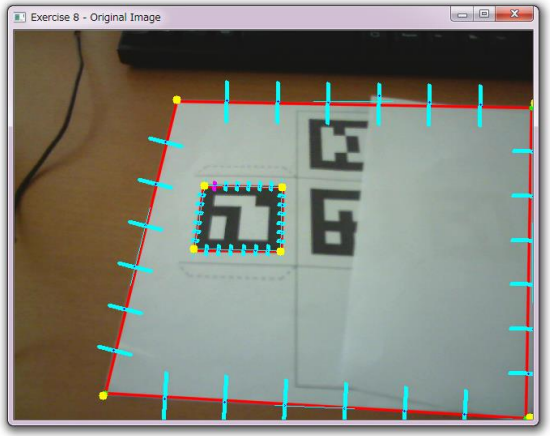
Ex. 7

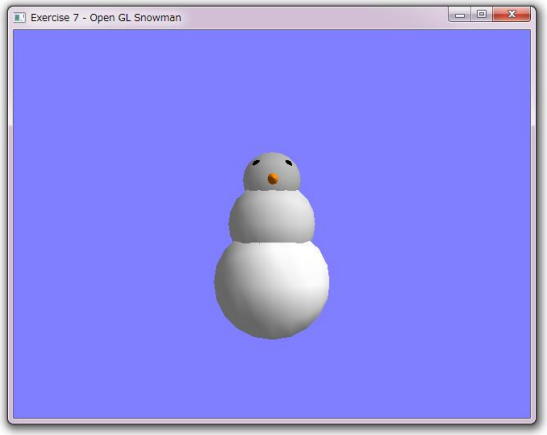


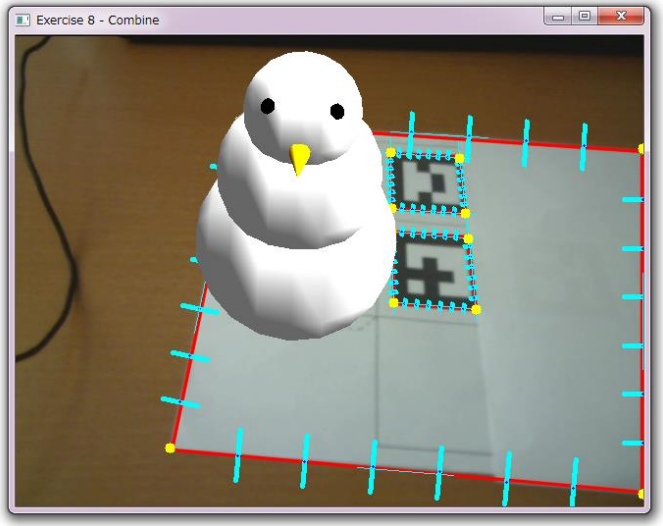
Today's Tutorial

Ex. 8

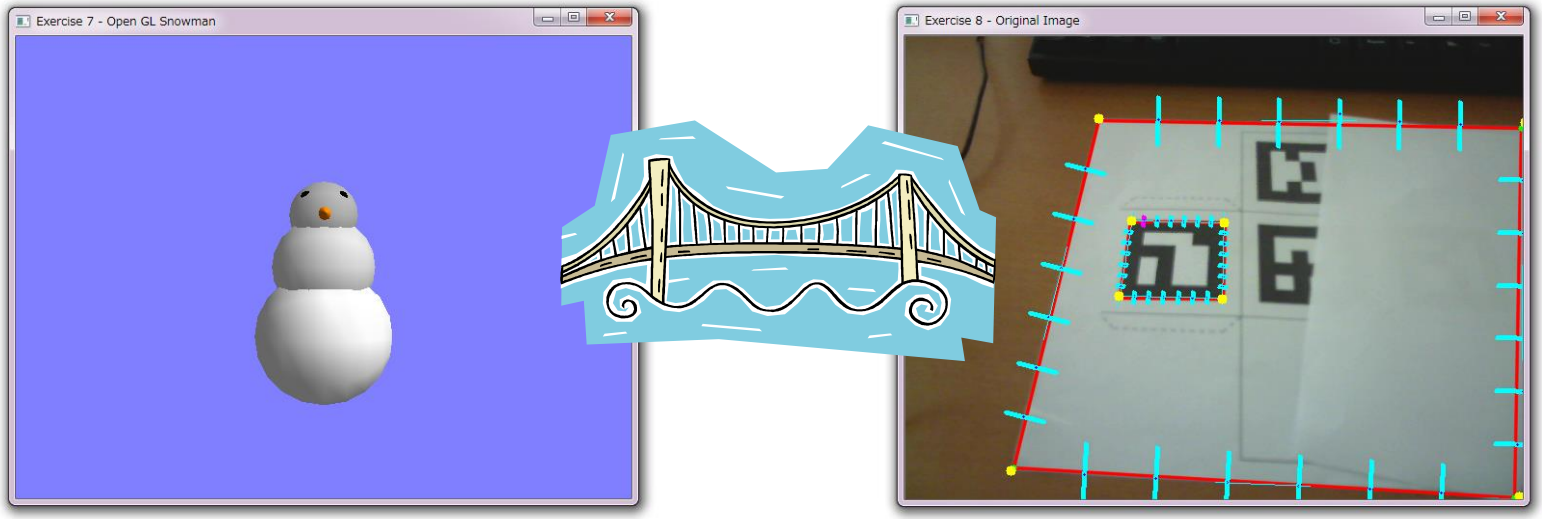
Combine AR







How to combine tracker and renderer?



Putting it All Together – Overview

Interprocess communication

Shared memory (fast)

Standard I/O: Pipe (slow)

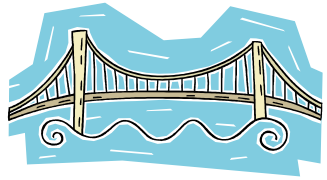
Network I/O: Sockets (slow)

File I/O

Interthread communication

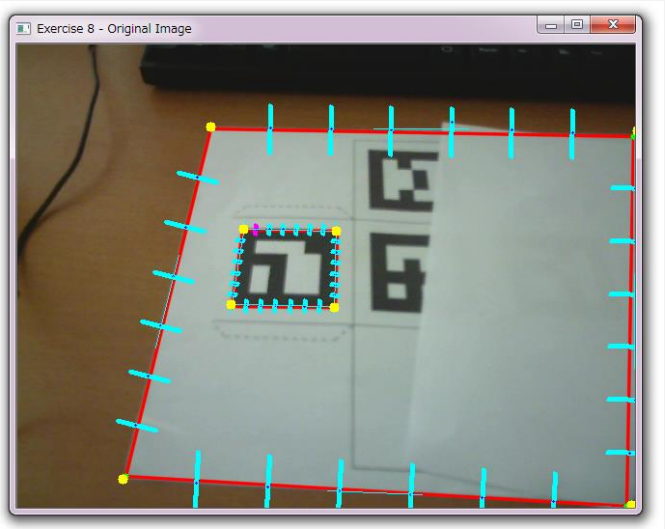
Easy and fast: Setup one program

Information that has to be conveyed



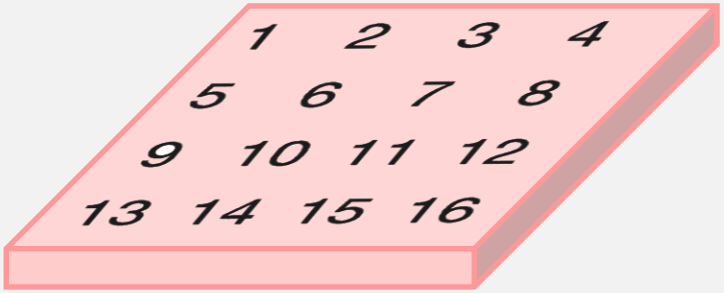
Image

(RGB) BGR image



Pose

4x4 Matrix



Main Program

```
// setup OpenCV
cv::Mat img_bgr;
initVideoStream(cap);
const double kMarkerSize = 0.048; // [m]
MarkerTracker markerTracker(kMarkerSize);

float resultMatrix[16];
/* Loop until the user closes the window */
while (!glfwWindowShouldClose(window))
{
    /* Capture here */
    cap >> img_bgr;

    if(img_bgr.empty()) { ... }

    /* Track a marker */
    markerTracker.findMarker( img_bgr, resultMatrix);

    /* Render here */
    display(window,          img_bgr, resultMatrix);
```

Sample class: “MarkerTracker.h”

Image

4x4 ModelView matrix

```
void findMarker( cv::Mat &img_bgr, float resultMatrix[16] );
```

Rendering the Background Image – Overview

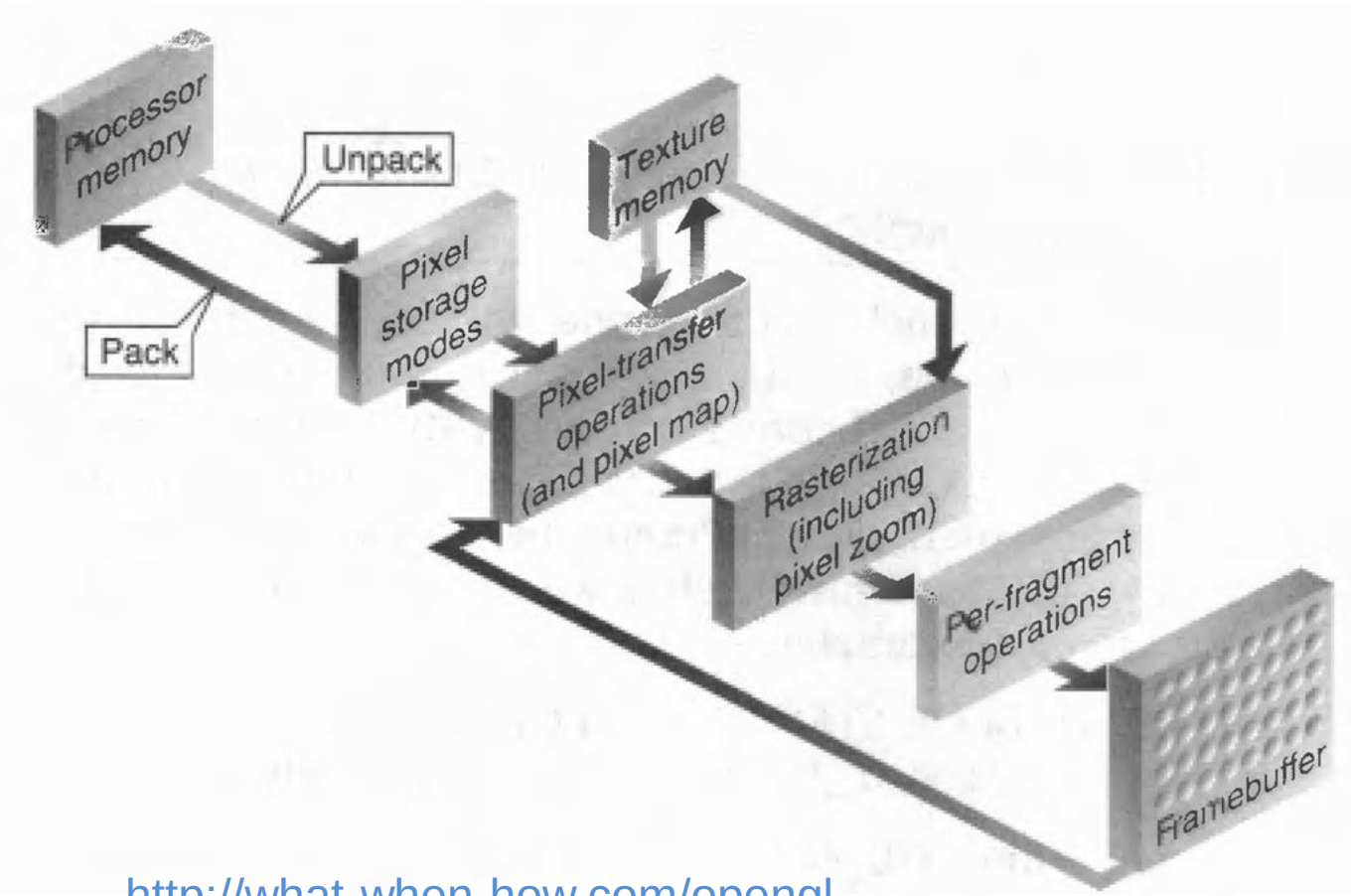
Once during initialization, tell OpenGL how to interprete our image data

- `glPixelStore`
- `glPixelZoom`

For each rendering cycle

- Store default projection matrix and disable the depth test
- `glDisable(), glMatrixMode(), glPushMatrix(), glLoadIdentity()`
- Set an orthogonal projection matrix covering the entire screen
- `glOrtho() with near = -1 and far = 1`
- Set the raster position
- `glRasterPos2()`
- Draw image using the pixel buffer from the IplImage.
- `glDrawPixels()`
- Restore the original projection matrix and re-enable the depth test.
- `glPopMatrix(), glEnable()`
- Load the modelview matrix and draw the snowman (as in last exercise)

Imaging Pipeline



http://what-when-how.com/opengl-programming-guide/imaging-pipeline-drawing-pixels-bitmaps-fonts-and-images-opengl-programming-part-1/

Rendering the Background Image – Setup Implementation Sketch

Initialize OpenGL pixel storage

```
/* program & OpenGL initialization */
void initGL(int argc, char *argv[])
{
    // initialize the GL library

// Added in Exercise 8 - End ************************************
    // pixel storage/packing stuff
    glPixelStorei( GL_PACK_ALIGNMENT,   1 ); // for glReadPixels
    glPixelStorei( GL_UNPACK_ALIGNMENT, 1 ); // for glTexImage2D
    glPixelZoom( 1.0, -1.0 );
// Added in Exercise 8 - End ************************************
```

```
unsigned char bkgnd[camera_width*camera_height*3];
```

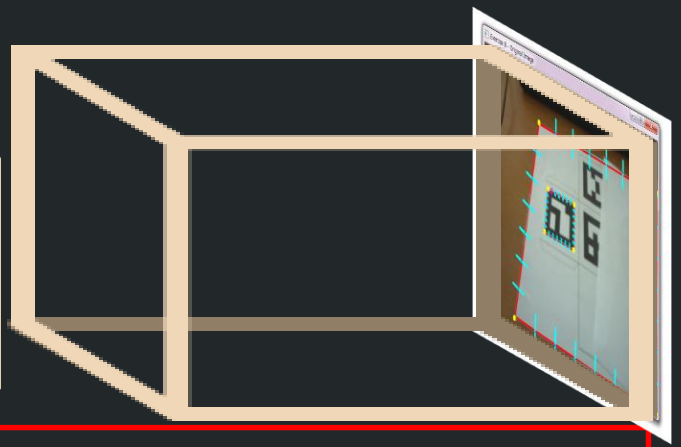
Rendering the Background Image – Renderer Implementation Sketch

```
unsigned char bkgnd[camera_width*camera_height*3];
```

```
void display( GLFWwindow* window, const cv::Mat &img_bgr )
{
// Added in Exercise 8 - Start ***********************************************
    memcpy( bkgnd, img_bgr.data, sizeof(bkgnd) );
// Added in Exercise 8 - End *************************************************

    int width0, height0;
    glfwGetFramebufferSize(window, &width0, &height0);
//  reshape(window, width, height);

    // clear buffers
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    // draw background image
    glDisable( GL_DEPTH_TEST );
```

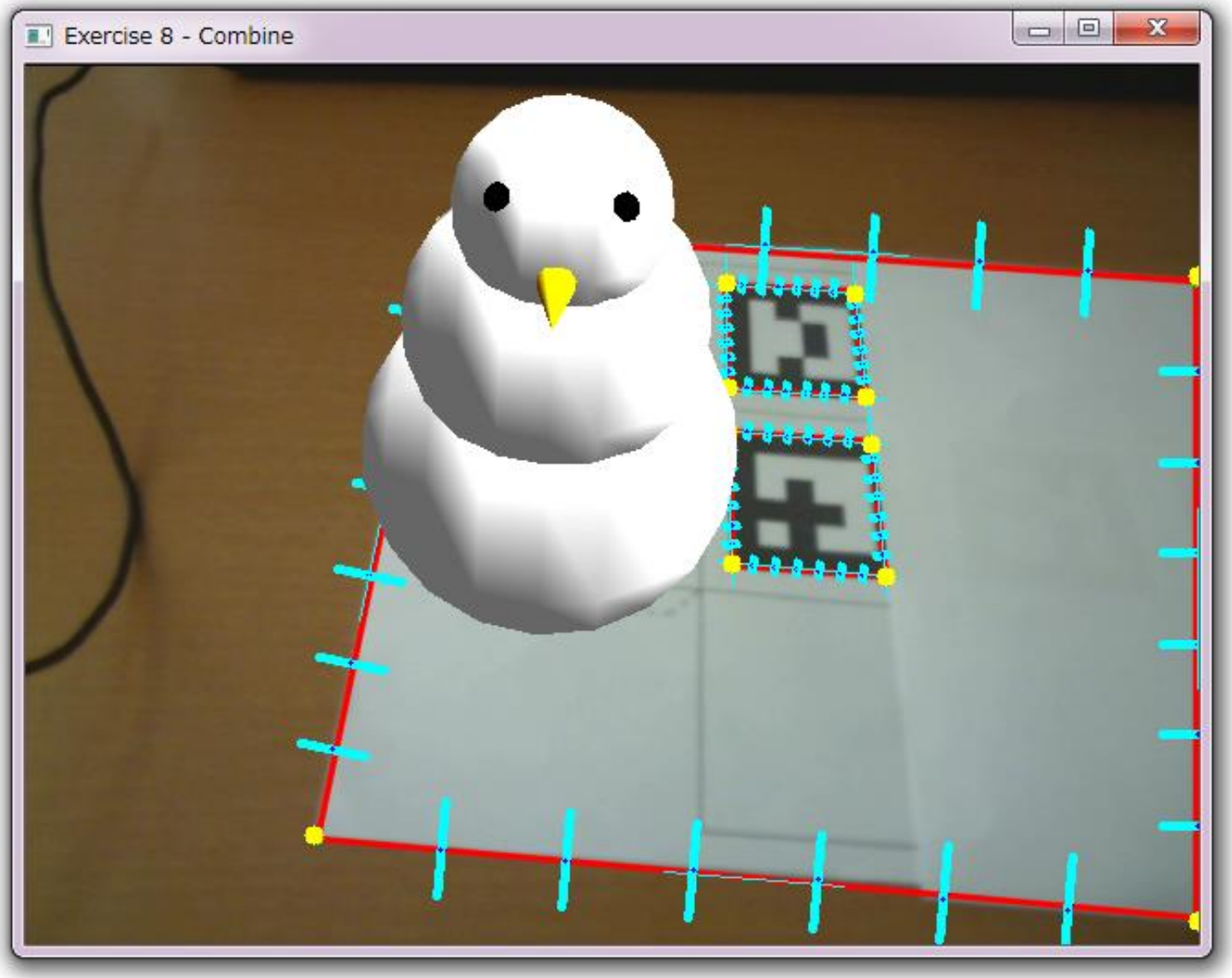
Rendering the Background Image – Renderer Implementation Sketch

```
glLoadIdentity();

// draw background image
glDisable( GL_DEPTH_TEST );

glMatrixMode( GL_PROJECTION );
glPushMatrix();
glLoadIdentity();
gluOrtho2D( 0.0, width, 0.0, height );

glRasterPos2i( 0, height-1 );
glDrawPixels( width, height, GL_BGR_EXT, GL_UNSIGNED_BYTE, bkgnd );

glPopMatrix();

glEnable(GL_DEPTH_TEST);
```



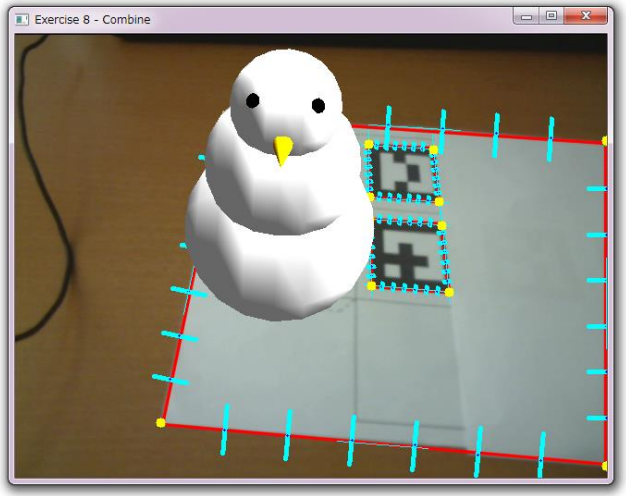
glOrtho(0.0, width, 0.0, height,-1,1);

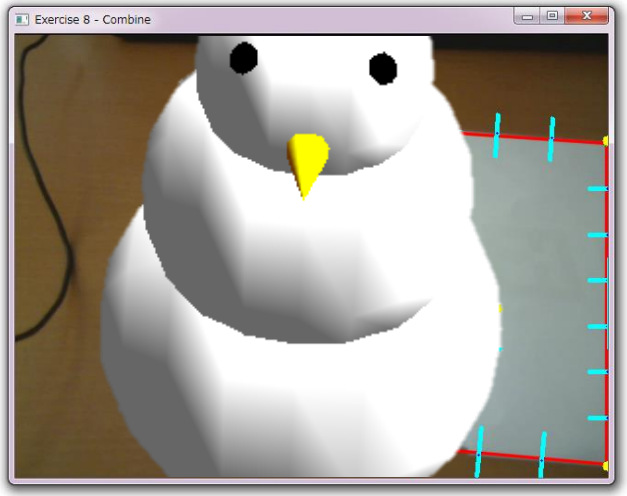
Result



Virtual and Real Camera - Equal FOV

```
// Note: Just setting the perspective is an easy hack.
// In fact, the camera should be calibrated.
// With such a calibration we would get the projection matrix.
// This matrix could then be loaded into GL_PROJECTION.
// If you are using another camera (which you'll do in most cases),
// you'll have to adjust the FOV
// value. How? Fiddle around: Move Marker to edge of display
// and check if you have to increase or decrease.
gluPerspective
    (virtual_camera_angle, ((double)width/(double)height), 0.01, 100 );

float ratio = width / height; float near = 0.01f, far = 100.f;
float top= tan((double)(fov*M_PI / 360.0f)) * near; float bottom= -top;
float left = ratio * bottom; float right = ratio * top;
glFrustum(left, right, bottom, top, near, far);
```





Homework

- Combine marker tracker and OpenGL rendering

Spoiler of the next tutorial

Spatial behavior

That‘s it…

- Questions