ARToolkit

CPS592 – Visual Computing and Mixed Reality

Introduction

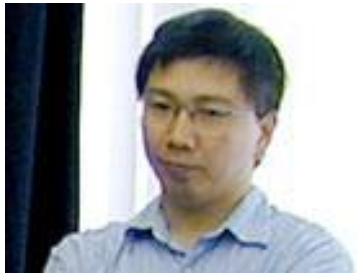
- ARToolKit is a **C and C++** language software library that lets programmers easily develop Augmented Reality applications.
- Recall: Augmented Reality (AR) is the overlay of virtual computer graphics images on the real world, and has many potential applications in industrial and academic research.
- Homepage: https://www.hitl.washington.edu/artoolkit

Problem

- One of the most difficult parts of developing an Augmented Reality application is precisely calculating the user's viewpoint in real time so that the virtual images are exactly aligned with real world objects.
- ARToolKit uses **computer vision techniques** to calculate the real camera position and orientation relative to marked cards (markers or patterns), allowing the programmer to overlay virtual objects onto these cards.

Author

- When Hirokazu Kato joined the Human Interface Technology Laboratory (HIT Lab) at the University of Washington as a visiting scholar in 1998, he started a research on Augmented Reality which is his main research topic.

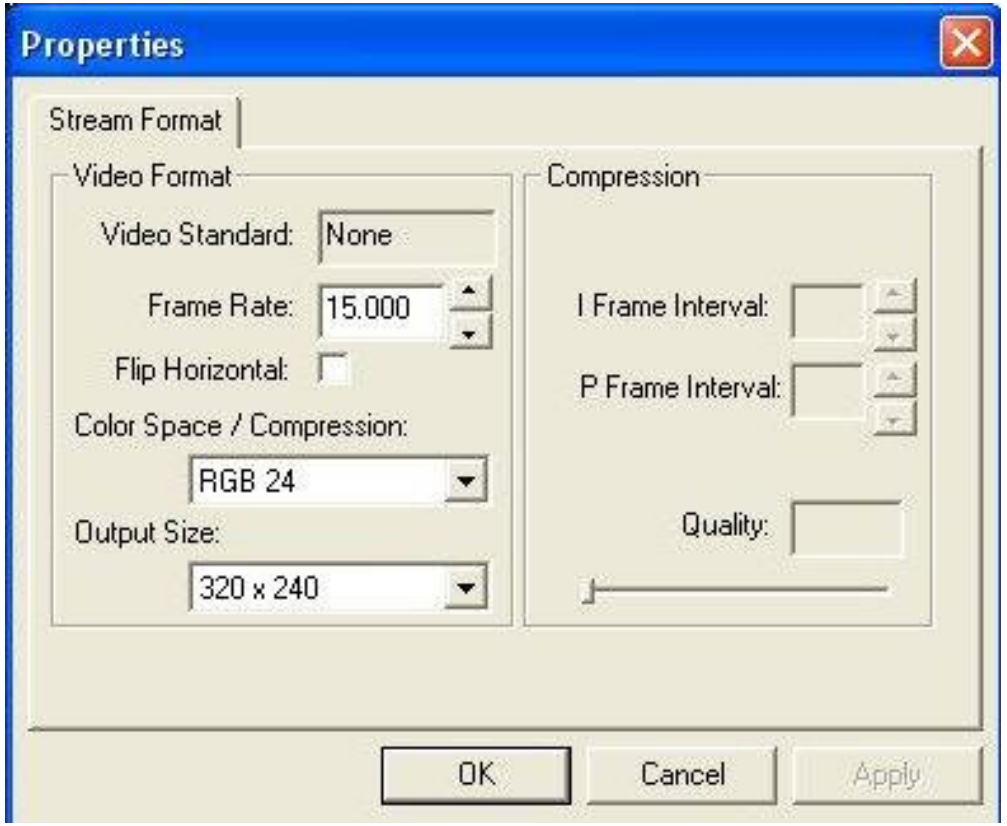


- Hirokazu Kato is currently a professor in the Nara Institute of Science and Technology (NAIST), Japan.

Kato in real life

Running ARToolKit

- On a Windows PC double click on the simple.exe icon in the bin directory from windows explorer

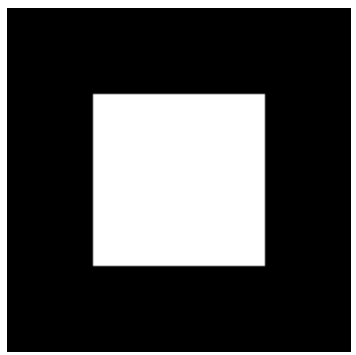
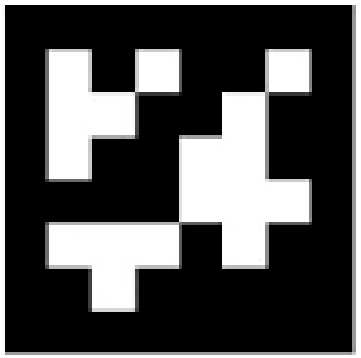


Running AR Toolkit

- If successful, a video window will appear on the screen. When you point the camera at the black square with HIRO printed in it you should see a virtual blue cube appear attached to the marker.
- As the real marker is moved the virtual object should move with it and appear exactly aligned with the marker.

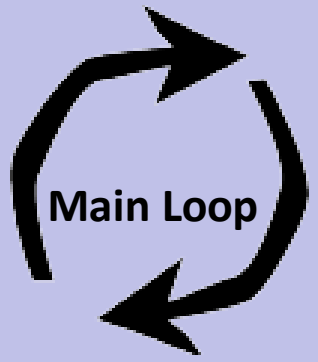
How ARToolkit works?

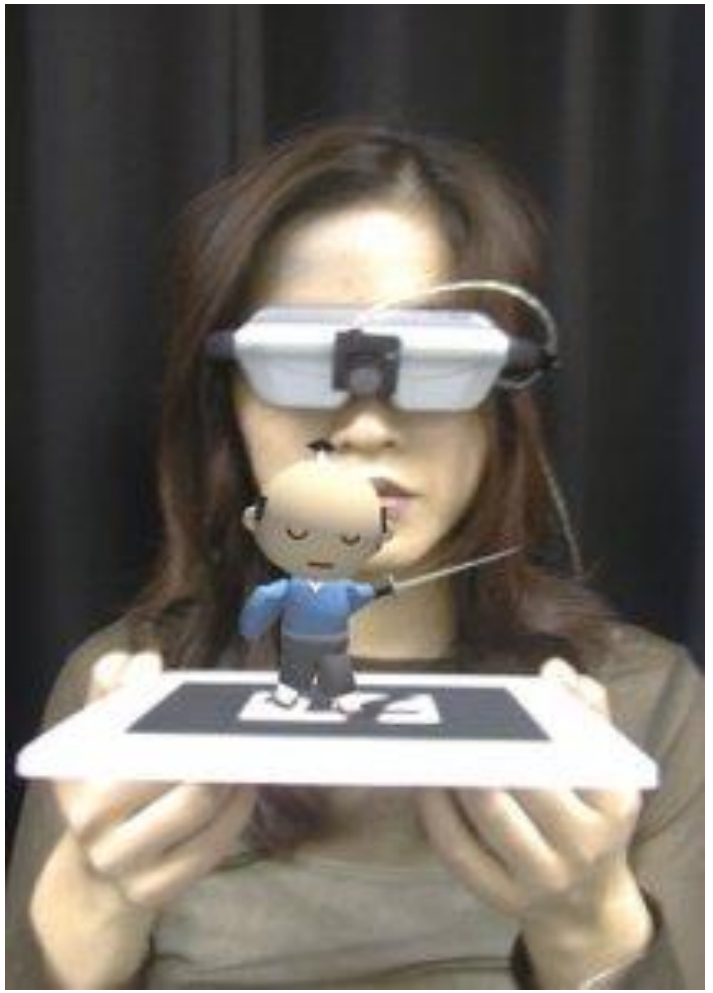
- The secret is in the black squares used as tracking markers.

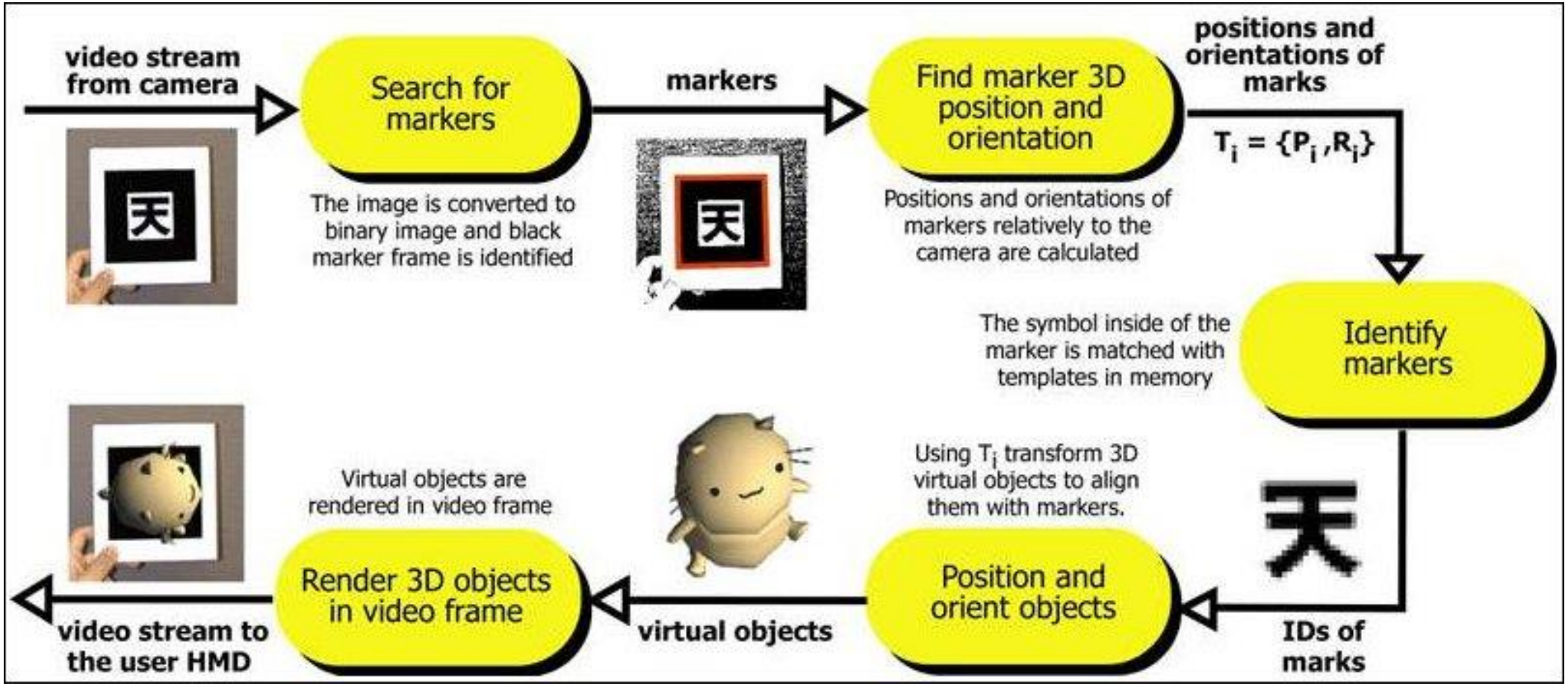
and many more …

How ARTookit works?

Initialization	1. Initialize the video capture and read in the marker pattern files and camera parameters.
	2. Grab a video input frame. 3. Detect the markers and recognized patterns in the video input frame. 4. Calculate the camera transformation relative to the detected patterns. 5. Draw the virtual objects on the detected patterns.
Shutdown	6. Close the video capture down.



The flowchart of ARTookit



Limitations

- Naturally the virtual objects will only appear when the tracking marks are in view. This may limit the size or movement of the virtual objects. It also means that if users cover up part of the pattern with their hands or other objects the virtual object will disappear.
- There are also *range issues*. The larger the physical pattern the further away the pattern can be detected and so the great volume the user can be tracked in.

Development with ARToolkit

ARToolKit Step	Functions
1. Initialize the application	init
2. Grab a video input frame	arVideoGetImage (called in mainLoop)
3. Detect the markers	arDetectMarker (called in mainLoop)
4. Calculate camera transformation	arGetTransMat (called in mainLoop)
5. Draw the virtual objects	draw (called in mainLoop)
6. Close the video capture down	cleanup

Main function

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    init();

    arVideoCapStart();
    argMainLoop( NULL, keyEvent, mainLoop );
    return (0);
}
```

Init

```
static void init( void )
{
   ARParam  wparam;

   /* open the video path */
   if( arVideoOpen( vconf ) < 0 ) exit(0);
   /* find the size of the window */
   if( arVideoInqSize(&xsize, &ysize) < 0 ) exit(0);
   printf("Image size (x,y) = (%d,%d)\n", xsize, ysize);

   /* set the initial camera parameters */
   if( arParamLoad(cparam_name, 1, &wparam) < 0 ) {
      printf("Camera parameter load error !!\n");
      exit(0);
   }

   arParamChangeSize( &wparam, xsize, ysize, &cparam );
   arInitCparam( &cparam );
   printf("*** Camera Parameter ***\n");
   arParamDisp( &cparam );

   if( (patt_id=arLoadPatt(patt_name)) < 0 ) {
      printf("pattern load error !!\n");
      exit(0);
   }

   /* open the graphics window */
   argInit( &cparam, 1.0, 0, 0, 0, 0 );
}
```

Main loop

```
static void mainLoop(void)
{
    ARUint8         *dataPtr;
    ARMarkerInfo    *marker_info;
    int             marker_num;
    int             j, k;

    /* grab a vide frame */
    if( (dataPtr = (ARUint8 *)arVideoGetImage()) == NULL ) {
        arUtilSleep(2);
        return;
    }
    if( count == 0 ) arUtilTimerReset();
    count++;

    argDrawMode2D();
    argDispImage( dataPtr, 0,0 );

    /* detect the markers in the video frame */
    if( arDetectMarker(dataPtr, thresh, &marker_info, &marker_num) < 0 ) {
        cleanup();
        exit(0);
    }

    arVideoCapNext();

    /* check for object visibility */
    k = -1;
    for( j = 0; j < marker_num; j++ ) {
        if( patt_id == marker_info[j].id ) {
            if( k == -1 ) k = j;
            else if( marker_info[k].cf < marker_info[j].cf ) k = j;
        }
    }
    if( k == -1 ) {
        argSwapBuffers();
        return;
    }

    /* get the transformation between the marker and the real camera */
    arGetTransMat(&marker_info[k], patt_center, patt_width, patt_trans);

      draw();

    argSwapBuffers();
}
```

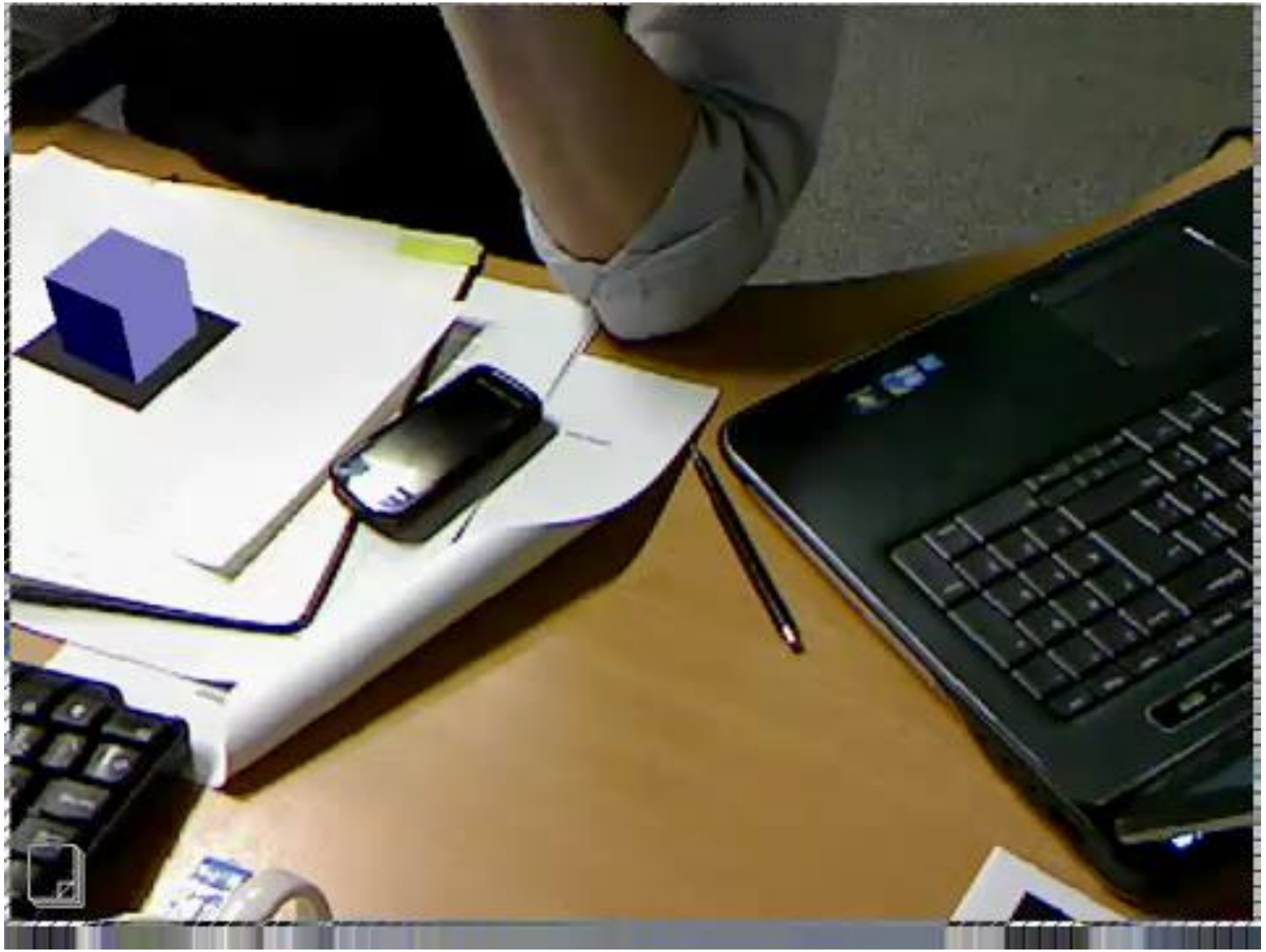
Draw

```
static void draw( void )
{
   double    gl_para[16];
   GLfloat   mat_ambient[]     = {0.0, 0.0, 1.0, 1.0};
   GLfloat   mat_flash[]       = {0.0, 0.0, 1.0, 1.0};
   GLfloat   mat_flash_shiny[] = {50.0};
   GLfloat   light_position[]  = {100.0,-200.0,200.0,0.0};
   GLfloat   ambi[]            = {0.1, 0.1, 0.1, 0.1};
   GLfloat   lightZeroColor[]  = {0.9, 0.9, 0.9, 0.1};

   argDrawMode3D();
   argDraw3dCamera( 0, 0 );
   glClearDepth( 1.0 );
   glClear(GL_DEPTH_BUFFER_BIT);
   glEnable(GL_DEPTH_TEST);
   glDepthFunc(GL_LEQUAL);

   /* load the camera transformation matrix */
   argConvGlpara(patt_trans, gl_para);
   glMatrixMode(GL_MODELVIEW);
   glLoadMatrixd( gl_para );

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);
   glLightfv(GL_LIGHT0, GL_AMBIENT, ambi);
   glLightfv(GL_LIGHT0, GL_DIFFUSE, lightZeroColor);
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_flash);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_flash_shiny);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
   glMatrixMode(GL_MODELVIEW);
   glTranslatef( 0.0, 0.0, 25.0 );
   glutSolidCube(50.0);
   glDisable( GL_LIGHTING );

   glDisable( GL_DEPTH_TEST );
}
```

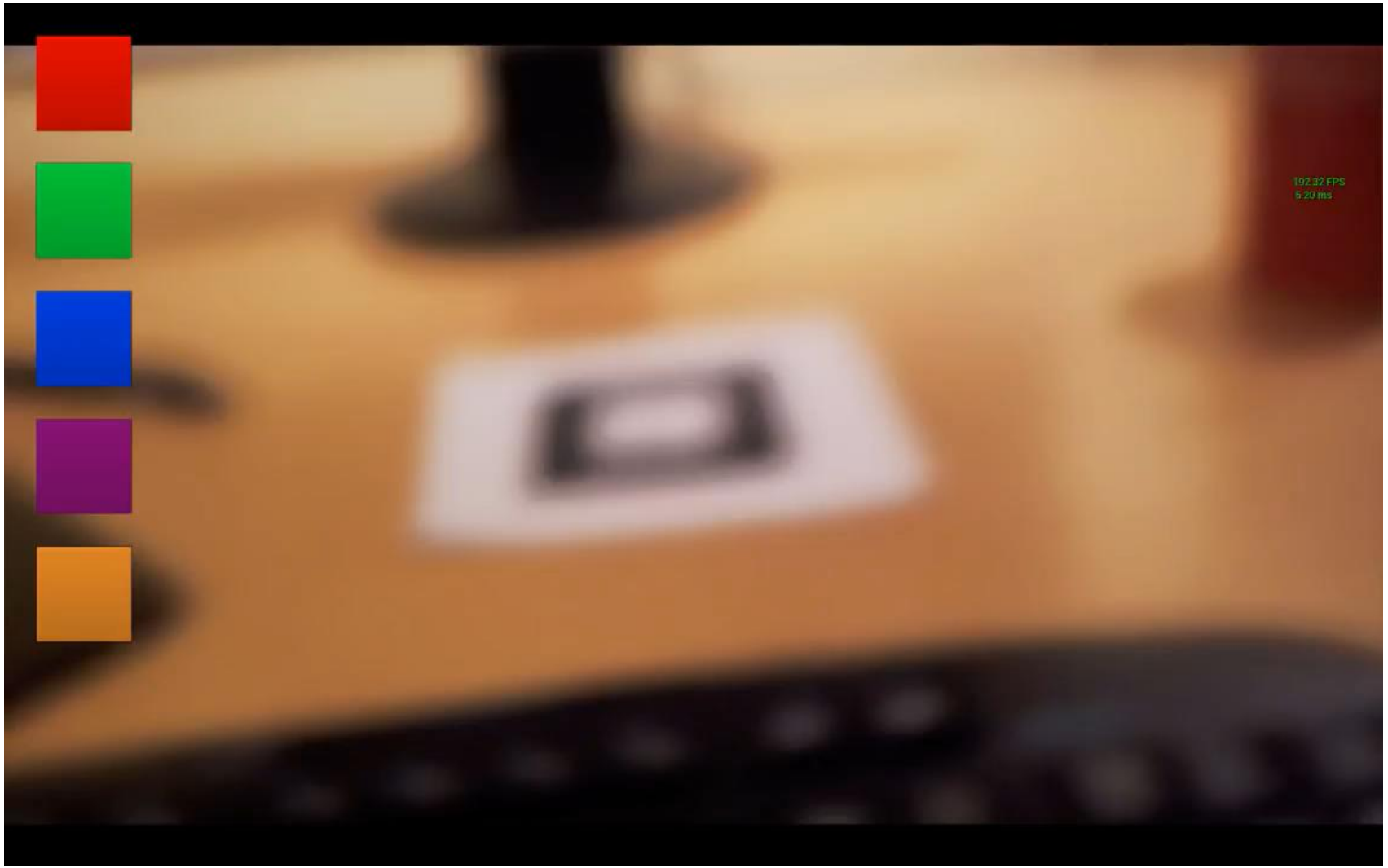
Clean up

```
/* cleanup function called when program exits
*/
static void cleanup(void)
{
        arVideoCapStop();
        arVideoClose();
        argCleanup();
}
```

Demo

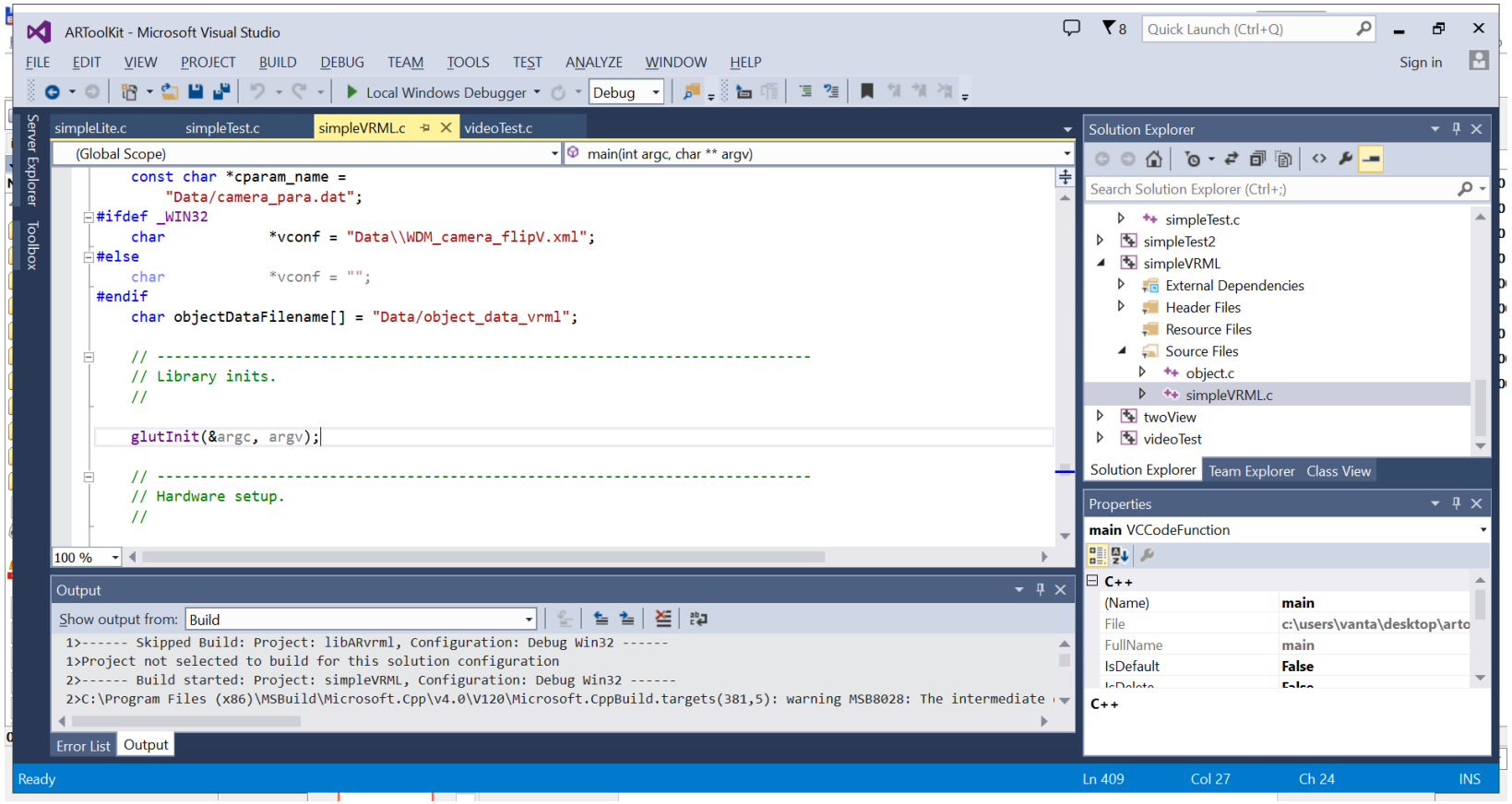


Another demo with different rendering library



For the next class

- Please install Visual Studio (version >= 2013)



Q&A