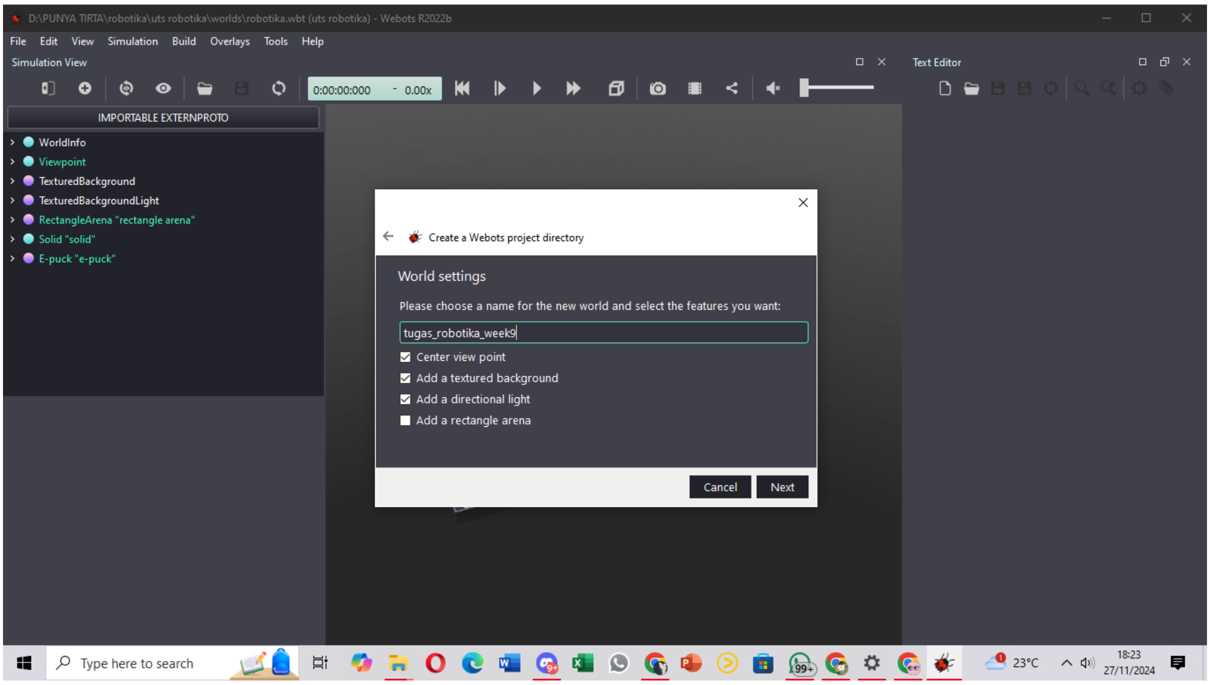
Ade Tirta Rahmat Hidayat – 1103203212 – Robotika TK45G06

DOKUMENTASI PEKERJAAN TUGAS WEEK 9 (CAMERA)

1. Membuat *new project*.



Create a Webots project directory

World settings

Please choose a name for the new world and select the features you want:

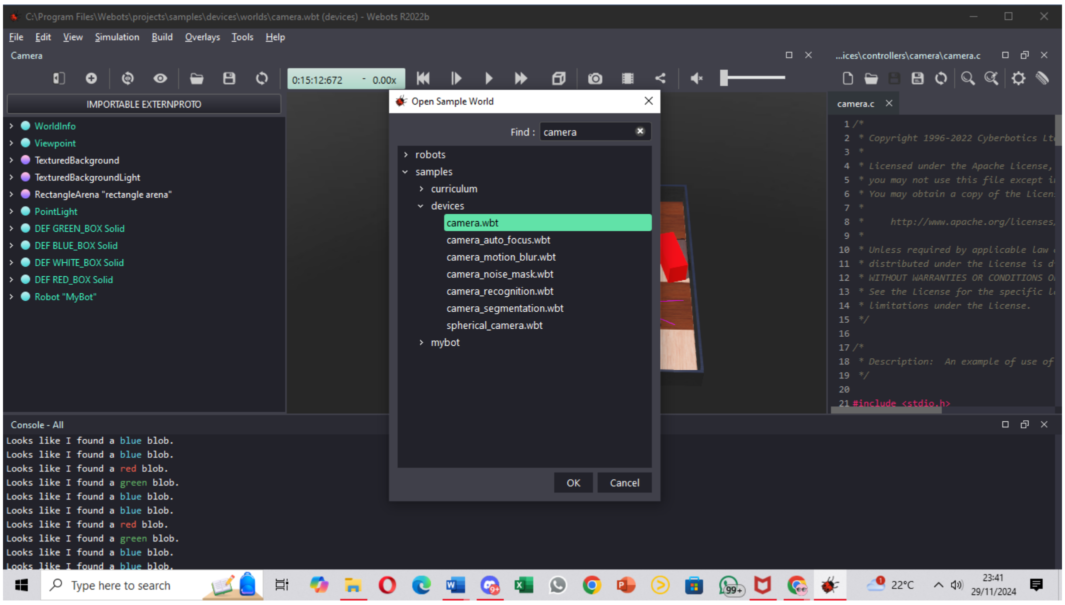
tugas_robotika_week9

☑ Center view point
☑ Add a textured background
☑ Add a directional light
☐ Add a rectangle arena

Cancel Next

2. Membuat *sample world* untuk implementasi *camera:*

   a. Camera robot untuk deteksi blob warna



Open Sample World

Find : camera

- robots
- samples
  - curriculum
  - devices
    - camera.wbt
    - camera_auto_focus.wbt
    - camera_motion_blur.wbt
    - camera_noise_mask.wbt
    - camera_recognition.wbt
    - camera_segmentation.wbt
    - spherical_camera.wbt
  - mybot

OK Cancel

```
/*
 * Copyright 1996-2022 Cyberbotics Lt
 *
 * Licensed under the Apache License,
 * you may not use this file except i
 * You may obtain a copy of the Licen
 *
 *     http://www.apache.org/licenses
 *
 * Unless required by applicable law
 * distributed under the License is d
 * WITHOUT WARRANTIES OR CONDITIONS O
 * See the License for the specific l
 * limitations under the License.
 */

/*
 * Description:  An example of use of
 */
#include <stdio.h>
```

```
Looks like I found a blue blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a blue blob.
```

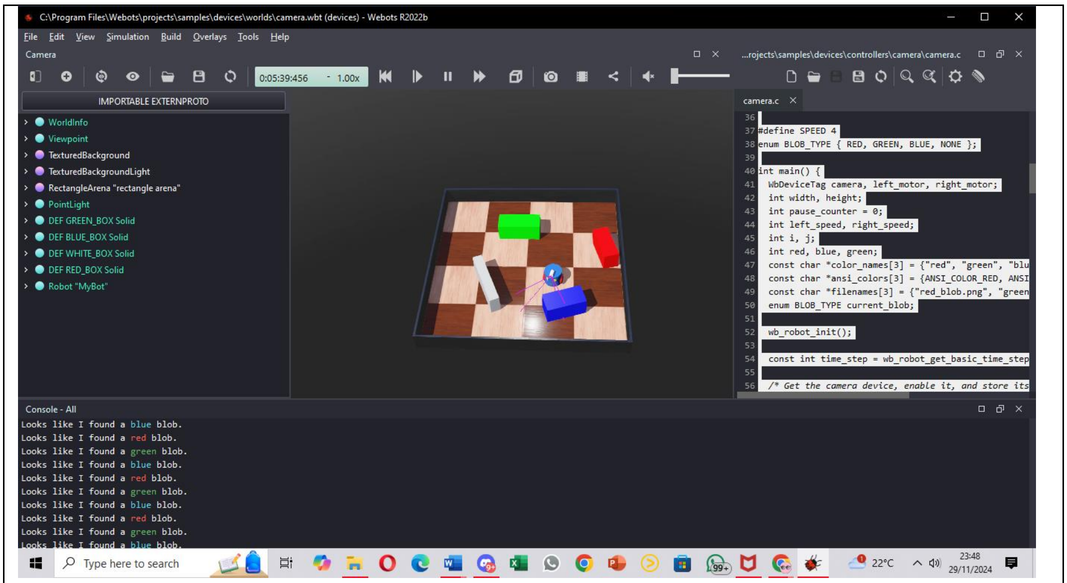
Code:

```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the
"License");
 * you may not use this file except in compliance with the
License.
```

```
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in
writing, software
 * distributed under the License is distributed on an "AS
IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
 * See the License for the specific language governing
permissions and
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera device.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <webots/camera.h>
#include <webots/motor.h>
#include <webots/robot.h>
#include <webots/utils/system.h>

#define ANSI_COLOR_RED "\x1b[31m"
#define ANSI_COLOR_GREEN "\x1b[32m"
#define ANSI_COLOR_YELLOW "\x1b[33m"
#define ANSI_COLOR_BLUE "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN "\x1b[36m"
#define ANSI_COLOR_RESET "\x1b[0m"

#define SPEED 4
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int width, height;
  int pause_counter = 0;
  int left_speed, right_speed;
  int i, j;
  int red, blue, green;
  const char *color_names[3] = {"red", "green", "blue"};
  const char *ansi_colors[3] =
{ANSI_COLOR_RED,ANSI_COLOR_GREEN, ANSI_COLOR_BLUE};
  const chr *filenames[3] = {"red_blob.png",
"green_blob.png", "blue_blob.png"};
 enum BLOB_TYPE current_blob;

  wb_robot_init();

  const int time_step = wb_robot_get_basic_time_step();
 /* Get the camera device, enable it, and store its width
and height */
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, time_step);
  width = wb_camera_get_width(camera);
  height = wb_camera_get_height(camera);

  /* get a handler to the motors and set target position
to infinity (speed control). */
  left_motor = wb_robot_get_device("left wheel motor");
  right_motor = wb_robot_get_device("right wheel motor");
  wb_motor_set_position(left_motor, INFINITY);
  wb_motor_set_position(right_motor, INFINITY);
  wb_motor_set_velocity(left_motor, 0.0);
  wb_motor_set_velocity(right_motor, 0.0);

  /* Main loop */
  while (wb_robot_step(time_step) != -1) {
    /* Get the new camera values */
    const unsigned char *image =
wb_camera_get_image(camera);

    /* Decrement the pause_counter */
    if (pause_counter > 0)
      pause_counter--;

    /*
     * Case 1
     * A blob was found recently
     * The robot waits in front of it until pause_counter
     * is decremented enough
     */
    if (pause_counter > 640 / time_step) {
      left_speed = 0;
      right_speed = 0;
    }
    /*
     * Case 2
     * A blob was found quite recently
     * The robot begins to turn but don't analyse theimage
for a while
     * otherwise the same blob would be found again
     */
    else if (pause_counter > 0) {
      left_speed = -SPEED;
      right_speed = SPEED;
    }
    /*
     * Case 3
     * The robot turns and analyse the camera image
inorder
     * to find a new blob
     */
    else if (!image) {  // image may be NULL
ifRobot.synchronization is FALSE
```

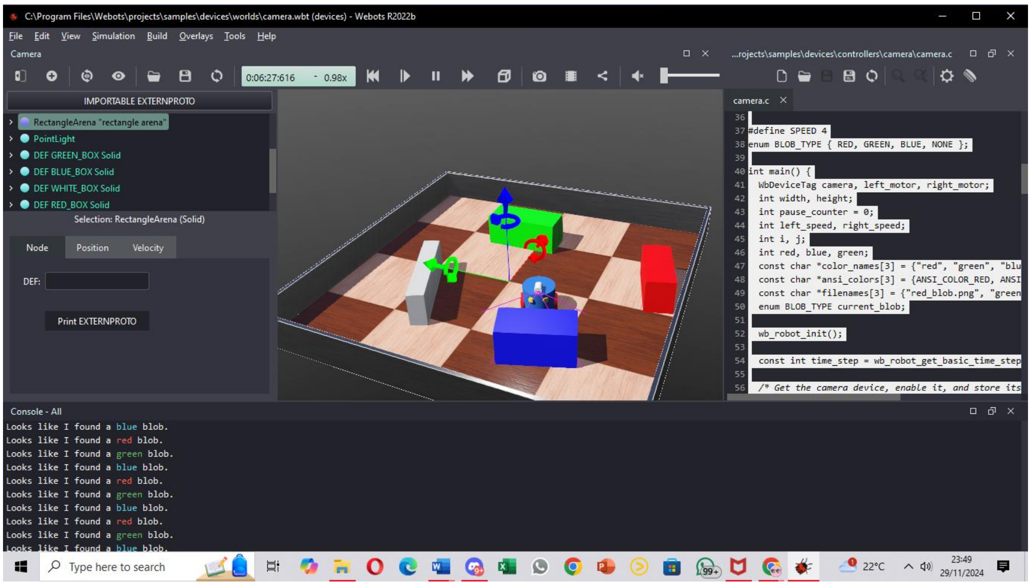
```
      left_speed = 0;
      right_speed = 0;
    } else {  // pause_counter == 0
      /* Reset the sums */
      red = 0;
      green = 0;
      blue = 0;

      /*
       * Here we analyse the image from the camera. The
goal is to detect a
       * blob (a spot of color) of a defined color in the
middle of our
       * screen.
       * In order to achieve that we simply parse the
image pixels of the
       * center of the image, and sum the color component
individually
       */
      for (i = width / 3; i < 2 * width / 3; i++) {
        for (j = height / 2; j < 3 * height / 4; j++) {
          red += wb_camera_image_get_red(image, width, i,
j);
           blue += wb_camera_image_get_blue(image, width,
     i, j);
           green += wb_camera_image_get_green(image, width,
     i, j);
            }
            }

        /*
             * If a component is much more represented than
     the other ones,
             * a blob is detected
             */
            if ((red > 3 * green) && (red > 3 * blue))
              current_blob = RED;
            else if ((green > 3 * red) && (green > 3 *
     blue))
              current_blob = GREEN;
            else if ((blue > 3 * red) && (blue > 3 *
     green))
              current_blob = BLUE;
            else
              current_blob = NONE;

            /*
             * Case 3a
             * No blob is detected
             * the robot continues to turn
             */
            if (current_blob == NONE) {
              left_speed = -SPEED;
              right_speed = SPEED;
            }
            /*
             * Case 3b
             * A blob is detected
             * the robot stops, stores the image, and
      changes its state
             */
            else {
              left_speed = 0;
              right_speed = 0;
              printf("Looks like I found a %s%s%s blob.\n",
      ansi_colors[current_blob], color_names[current_blob],
      ANSI_COLOR_RESET);
              // compute the file path in the user
      directory
              char *filepath;
      #ifdef _WIN32
              const char *user_directory =
      wbu_system_short_path(wbu_system_getenv("USERPROFILE"
      ));
              filepath = (char
      *)malloc(strlen(user_directory) + 16);
              strcpy(filepath, user_directory);
              strcat(filepath, "\\");
      #else
              const char *user_directory =
      wbu_system_getenv("HOME");
              filepath = (char
      *)malloc(strlen(user_directory) + 16);
              strcpy(filepath, user_directory);
              strcat(filepath, "/");
      #endif
              strcat(filepath, filenames[current_blob]);
              wb_camera_save_image(camera, filepath, 100);
              free(filepath);
              pause_counter = 1280 / time_step;
            }
          }

          /* Set the motor speeds. */
          wb_motor_set_velocity(left_motor, left_speed);
          wb_motor_set_velocity(right_motor, right_speed);
        }

        wb_robot_cleanup();

        return 0;
}
```

Output:



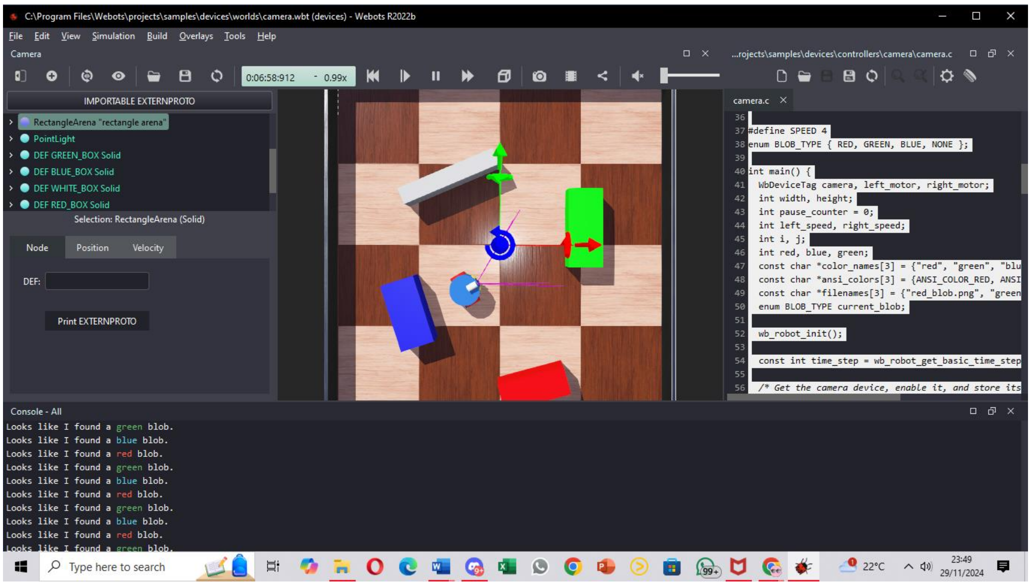
```

#define SPEED 4
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int width, height;
  int pause_counter = 0;
  int left_speed, right_speed;
  int i, j;
  int red, blue, green;
  const char *color_names[3] = {"red", "green", "blu
  const char *ansi_colors[3] = {ANSI_COLOR_RED, ANSI
  const char *filenames[3] = {"red_blob.png", "green
  enum BLOB_TYPE current_blob;

  wb_robot_init();

  const int time_step = wb_robot_get_basic_time_step

  /* Get the camera device, enable it, and store its
```

```
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
```


C:\Program Files\Webots\projects\samples\devices\worlds\camera.wbt (devices) - Webots R2022b
File Edit View Simulation Build Overlays Tools Help
Camera
0:06:27:616
0.98x
IMPORTABLE EXTERNPROTO
RectangleArena "rectangle arena"
PointLight
DEF GREEN_BOX Solid
DEF BLUE_BOX Solid
DEF WHITE_BOX Solid
DEF RED_BOX Solid
Selection: RectangleArena (Solid)
Node
Position
Velocity
DEF:
Print EXTERNPROTO
...rojects\samples\devices\controllers\camera\camera.c
camera.c
Console - All
Type here to search
22°C
23:49
29/11/2024

```

#define SPEED 4
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int width, height;
  int pause_counter = 0;
  int left_speed, right_speed;
  int i, j;
  int red, blue, green;
  const char *color_names[3] = {"red", "green", "blu
  const char *ansi_colors[3] = {ANSI_COLOR_RED, ANSI
  const char *filenames[3] = {"red_blob.png", "green
  enum BLOB_TYPE current_blob;

  wb_robot_init();

  const int time_step = wb_robot_get_basic_time_step

  /* Get the camera device, enable it, and store its
```

```
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
```


C:\Program Files\Webots\projects\samples\devices\worlds\camera.wbt (devices) - Webots R2022b
File Edit View Simulation Build Overlays Tools Help
Camera
0:06:58:912
0.99x
IMPORTABLE EXTERNPROTO
RectangleArena "rectangle arena"
PointLight
DEF GREEN_BOX Solid
DEF BLUE_BOX Solid
DEF WHITE_BOX Solid
DEF RED_BOX Solid
Selection: RectangleArena (Solid)
Node
Position
Velocity
DEF:
Print EXTERNPROTO
...rojects\samples\devices\controllers\camera\camera.c
camera.c
Console - All
Type here to search
22°C
23:49
29/11/2024

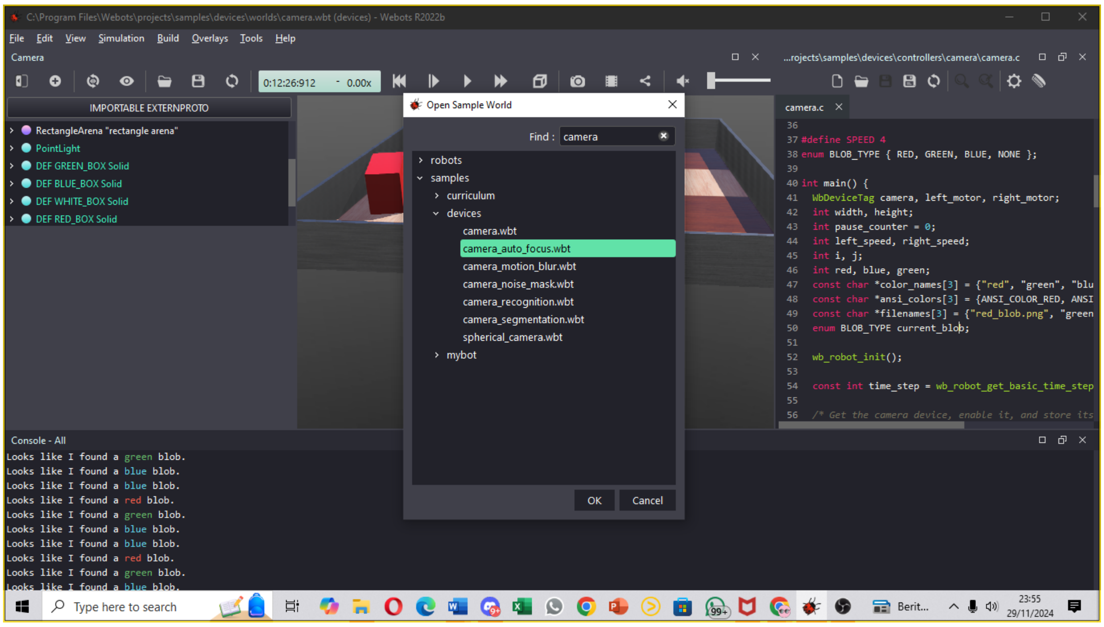
```

#define SPEED 4
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int width, height;
  int pause_counter = 0;
  int left_speed, right_speed;
  int i, j;
  int red, blue, green;
  const char *color_names[3] = {"red", "green", "blu
  const char *ansi_colors[3] = {ANSI_COLOR_RED, ANSI
  const char *filenames[3] = {"red_blob.png", "green
  enum BLOB_TYPE current_blob;

  wb_robot_init();

  const int time_step = wb_robot_get_basic_time_step

  /* Get the camera device, enable it, and store its
```

```
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
```

Link Video Output:

https://drive.google.com/file/d/1yJWTezrfJPFb8UwJ5mk_PuRKDwe2CLQZ/view?usp=drive_link

b. Camera robot dengan fokus berdasarkan objek yang ada di depannya



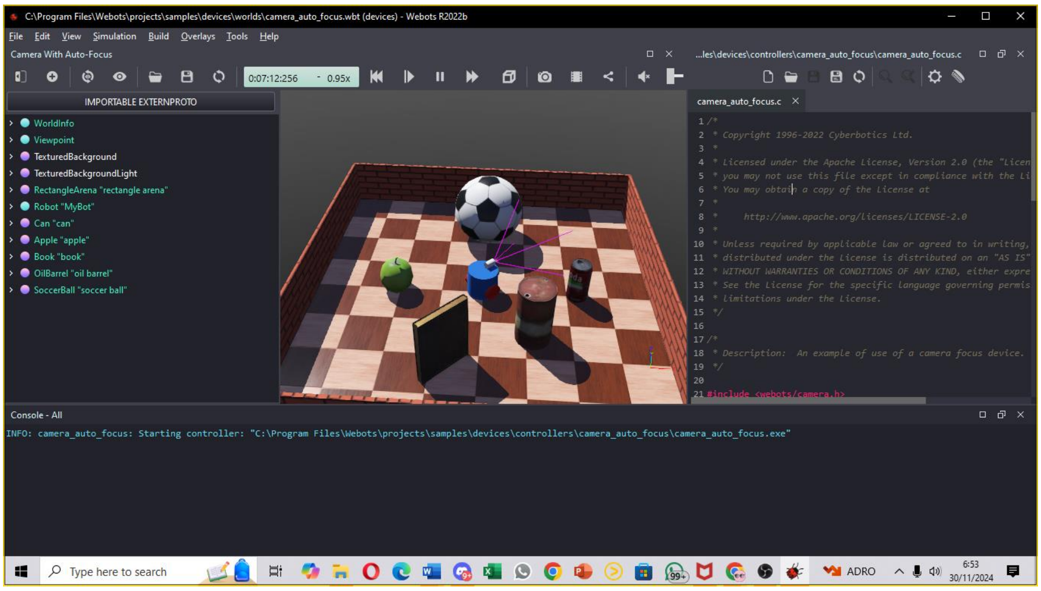
```
#define SPEED 4
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int width, height;
  int pause_counter = 0;
  int left_speed, right_speed;
  int i, j;
  int red, blue, green;
  const char *color_names[3] = {"red", "green", "blu
  const char *ansi_colors[3] = {ANSI_COLOR_RED, ANSI
  const char *filenames[3] = {"red_blob.png", "green
  enum BLOB_TYPE current_blob;

  wb_robot_init();

  const int time_step = wb_robot_get_basic_time_step

  /* Get the camera device, enable it, and store its
```

```
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
```

Code:

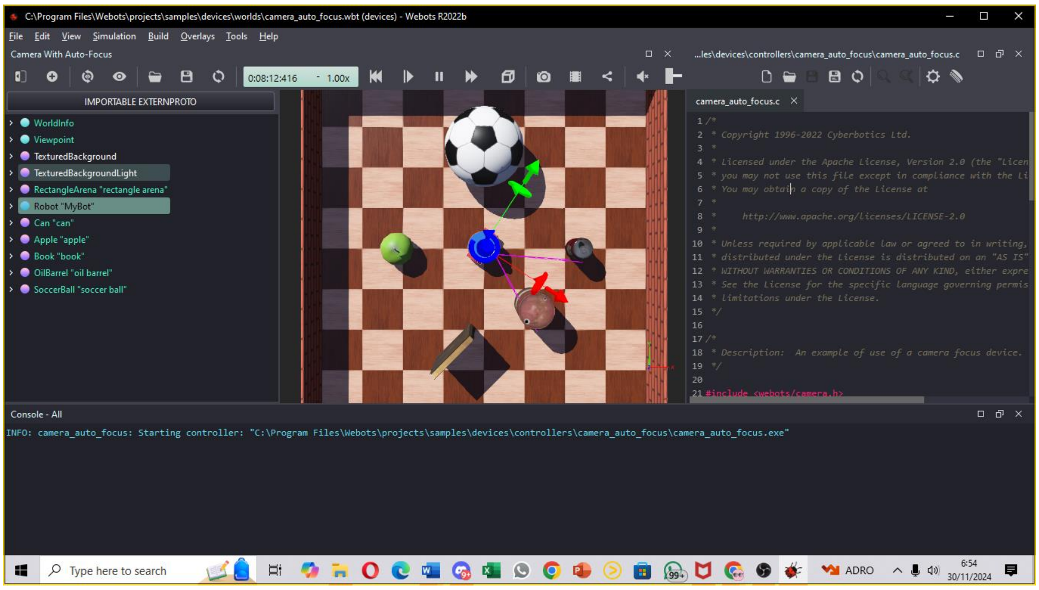
```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the
"License");
 * you may not use this file except in compliance with the
License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in
writing, software
 * distributed under the License is distributed on an "AS
IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
 * See the License for the specific language governing
permissions and
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera focus
device.
 */

#include <webots/camera.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>
#include <webots/robot.h>

#define SPEED 1
#define TIME_STEP 32

int main() {
  WbDeviceTag camera, distance_sensor, left_motor,
right_motor;

  wb_robot_init();

  /* Get the camera device, enable it */
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, TIME_STEP);

  /* Get the camera device, enable it */
  distance_sensor = wb_robot_get_device("distance
sensor");
  wb_distance_sensor_enable(distance_sensor, TIME_STEP);

  /* get a handler to the motors and set target position
to infinity (speed control). */
  left_motor = wb_robot_get_device("left wheel motor");
  right_motor = wb_robot_get_device("right wheel motor");
  wb_motor_set_position(left_motor, INFINITY);
  wb_motor_set_position(right_motor, INFINITY);
  wb_motor_set_velocity(left_motor, 0.0);
  wb_motor_set_velocity(right_motor, 0.0);

  /* Set the motors speed */
  wb_motor_set_velocity(left_motor, -SPEED);
  wb_motor_set_velocity(right_motor, SPEED);

  /* Main loop */
  while (wb_robot_step(TIME_STEP) != -1) {
    const double object_distance =
wb_distance_sensor_get_value(distance_sensor) / 1000;
    wb_camera_set_focal_distance(camera, object_distance);
  }

  wb_robot_cleanup();

  return 0;
}
/*
```

Output:



```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "Licen
 * you may not use this file except in compliance with the Li
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * distributed under the License is distributed on an "AS IS"
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expre
 * See the License for the specific language governing permis
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera focus device.
 */

#include <webots/camera.h>
```

```
INFO: camera_auto_focus: Starting controller: "C:\Program Files\Webots\projects\samples\devices\controllers\camera_auto_focus\camera_auto_focus.exe"
```



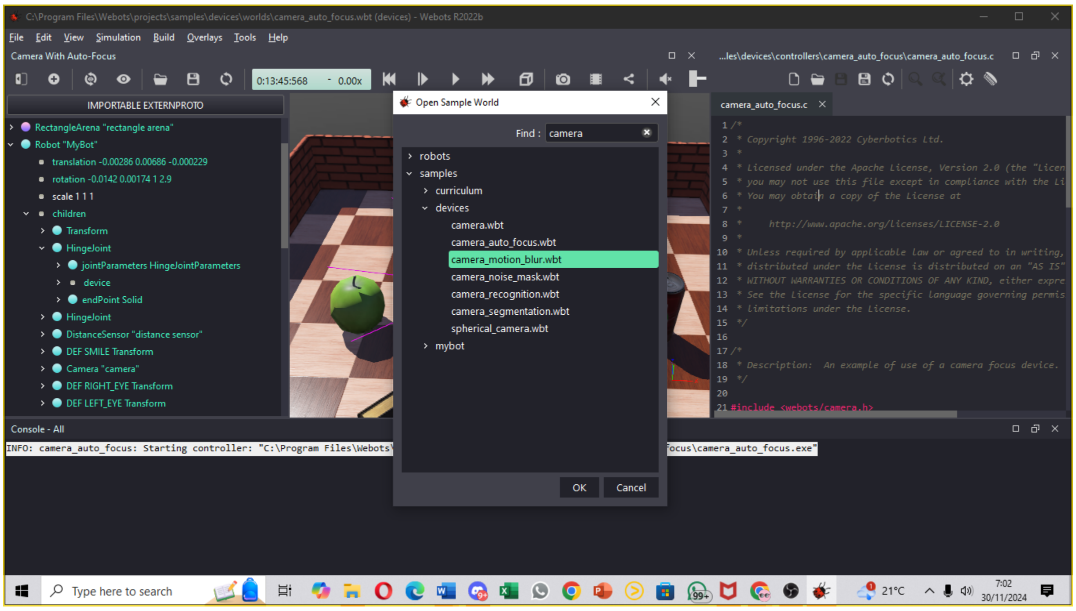
```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "Licen
 * you may not use this file except in compliance with the Li
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * distributed under the License is distributed on an "AS IS"
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expre
 * See the License for the specific language governing permis
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera focus device.
 */

#include <webots/camera.h>
```

```
INFO: camera_auto_focus: Starting controller: "C:\Program Files\Webots\projects\samples\devices\controllers\camera_auto_focus\camera_auto_focus.exe"
```

Link Video Output:

https://drive.google.com/file/d/1vGKCUKuwPswfC2l1yU6J-3SkNaKfQorG/view?usp=sharing

c. Camera robot deteksi blob berwarna pada robot dengan efek *motion blur camera*



```
RectangleArena "rectangle arena"
Robot "MyBot"
    translation -0.00286 0.00686 -0.000229
    rotation -0.0142 0.00174 1 2.9
    scale 1 1 1
    children
        Transform
        HingeJoint
            jointParameters HingeJointParameters
            device
            endPoint Solid
        HingeJoint
        DistanceSensor "distance sensor"
        DEF SMILE Transform
        Camera "camera"
        DEF RIGHT_EYE Transform
        DEF LEFT_EYE Transform
```

```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "Licen
 * you may not use this file except in compliance with the Li
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * distributed under the License is distributed on an "AS IS"
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expre
 * See the License for the specific language governing permis
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera focus device.
 */

#include <webots/camera.h>
```

```
INFO: camera_auto_focus: Starting controller: "C:\Program Files\Webots\ocus\camera_auto_focus.exe"
```

Code:

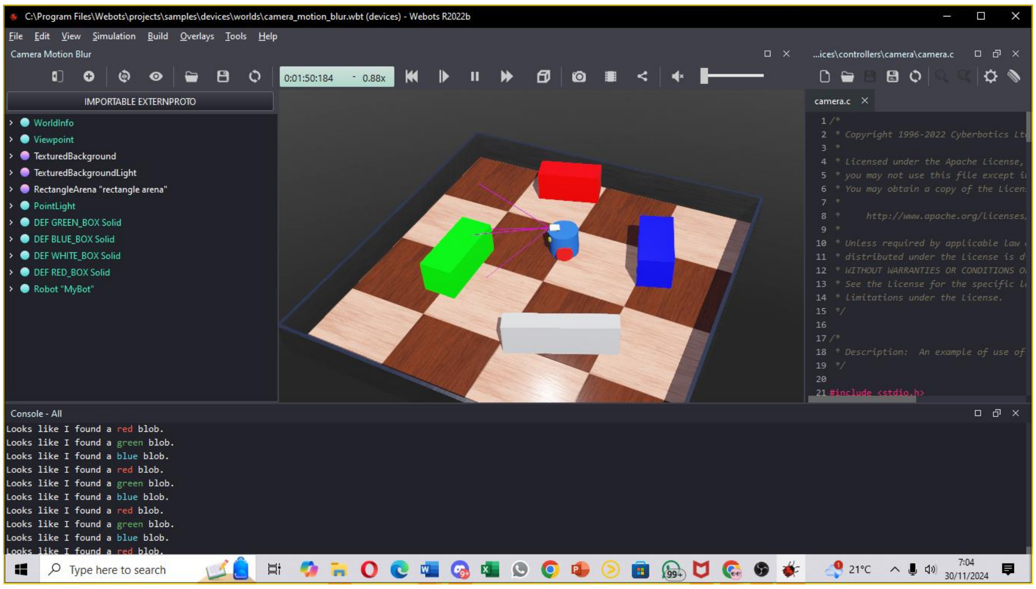
```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0
(the "License");
 * you may not use this file except in compliance
with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in
writing, software
 * distributed under the License is distributed on an
"AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
either express or implied.
 * See the License for the specific language
governing permissions and
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera
device.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <webots/camera.h>
#include <webots/motor.h>
#include <webots/robot.h>
#include <webots/utils/system.h>
```

```
#define ANSI_COLOR_RED "\x1b[31m"
#define ANSI_COLOR_GREEN "\x1b[32m"
#define ANSI_COLOR_YELLOW "\x1b[33m"
#define ANSI_COLOR_BLUE "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN "\x1b[36m"
#define ANSI_COLOR_RESET "\x1b[0m"

#define SPEED 4
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int width, height;
  int pause_counter = 0;
  int left_speed, right_speed;
  int i, j;
  int red, blue, green;
  const char *color_names[3] = {"red", "green",
"blue"};
  const char *ansi_colors[3] = {ANSI_COLOR_RED,
ANSI_COLOR_GREEN, ANSI_COLOR_BLUE};
  const char *filenames[3] = {"red_blob.png",
"green_blob.png", "blue_blob.png"};
  enum BLOB_TYPE current_blob;

  wb_robot_init();

  const int time_step =
wb_robot_get_basic_time_step();

  /* Get the camera device, enable it, and store its
width and height */
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, time_step);
  width = wb_camera_get_width(camera);
  height = wb_camera_get_height(camera);

  /* get a handler to the motors and set target
position to infinity (speed control). */
  left_motor = wb_robot_get_device("left wheel
motor");
  right_motor = wb_robot_get_device("right wheel
motor");
  wb_motor_set_position(left_motor, INFINITY);
  wb_motor_set_position(right_motor, INFINITY);
  wb_motor_set_velocity(left_motor, 0.0);
  wb_motor_set_velocity(right_motor, 0.0);

  /* Main loop */
  while (wb_robot_step(time_step) != -1) {
    /* Get the new camera values */
    const unsigned char *image =
wb_camera_get_image(camera);
```

```
    /* Decrement the pause_counter */
    if (pause_counter > 0)
      pause_counter--;

    /*
     * Case 1
     * A blob was found recently
     * The robot waits in front of it until
pause_counter
     * is decremented enough
     */
    if (pause_counter > 640 / time_step) {
      left_speed = 0;
      right_speed = 0;
    }
    /*
     * Case 2
     * A blob was found quite recently
     * The robot begins to turn but don't analyse the
image for a while,
     * otherwise the same blob would be found again
     */
    else if (pause_counter > 0) {
      left_speed = -SPEED;
      right_speed = SPEED;
    }
    /*
     * Case 3
     * The robot turns and analyse the camera image
in order
     * to find a new blob
     */
    else if (!image) {  // image may be NULL if
Robot.synchronization is FALSE
      left_speed = 0;
      right_speed = 0;
    } else {  // pause_counter == 0
      /* Reset the sums */
      red = 0;
      green = 0;
      blue = 0;

      /*
       * Here we analyse the image from the camera.
The goal is to detect a
       * blob (a spot of color) of a defined color in
the middle of our
       * screen.
       * In order to achieve that we simply parse the
image pixels of the
       * center of the image, and sum the color
components individually
       */
      for (i = width / 3; i < 2 * width / 3; i++) {
        for (j = height / 2; j < 3 * height / 4; j++)
{
          red += wb_camera_image_get_red(image,
width, i, j);
          blue += wb_camera_image_get_blue(image,
width, i, j);
          green += wb_camera_image_get_green(image,
width, i, j);
        }
      }

      /*
       * If a component is much more represented than
the other ones,
       * a blob is detected
       */
      if ((red > 3 * green) && (red > 3 * blue))
        current_blob = RED;
      else if ((green > 3 * red) && (green > 3 *
blue))
        current_blob = GREEN;
      else if ((blue > 3 * red) && (blue > 3 *
green))
        current_blob = BLUE;
      else
        current_blob = NONE;

      /*
       * Case 3a
       * No blob is detected
       * the robot continues to turn
       */
      if (current_blob == NONE) {
        left_speed = -SPEED;
        right_speed = SPEED;
      }
      /*
       * Case 3b
       * A blob is detected
       * the robot stops, stores the image, and
changes its state
       */
      else {
        left_speed = 0;
        right_speed = 0;
        printf("Looks like I found a %s%s%s blob.\n",
ansi_colors[current_blob], color_names[current_blob],
ANSI_COLOR_RESET);
        // compute the file path in the user
directory
        char *filepath;
#ifdef _WIN32
        const char *user_directory =
wbu_system_short_path(wbu_system_getenv("USERPROFILE"
));
```

```
       filepath = (char
   *)malloc(strlen(user_directory) + 16);
       strcpy(filepath, user_directory);
       strcat(filepath, "\\");
   #else
       const char *user_directory =
   wbu_system_getenv("HOME");
       filepath = (char
   *)malloc(strlen(user_directory) + 16);
       strcpy(filepath, user_directory);
       strcat(filepath, "/");
   #endif
       strcat(filepath, filenames[current_blob]);
       wb_camera_save_image(camera, filepath, 100);
       free(filepath);
       pause_counter = 1280 / time_step;
      }
     }

     /* Set the motor speeds. */
     wb_motor_set_velocity(left_motor, left_speed);
     wb_motor_set_velocity(right_motor, right_speed);
    }

    wb_robot_cleanup();

    return 0;
}
```

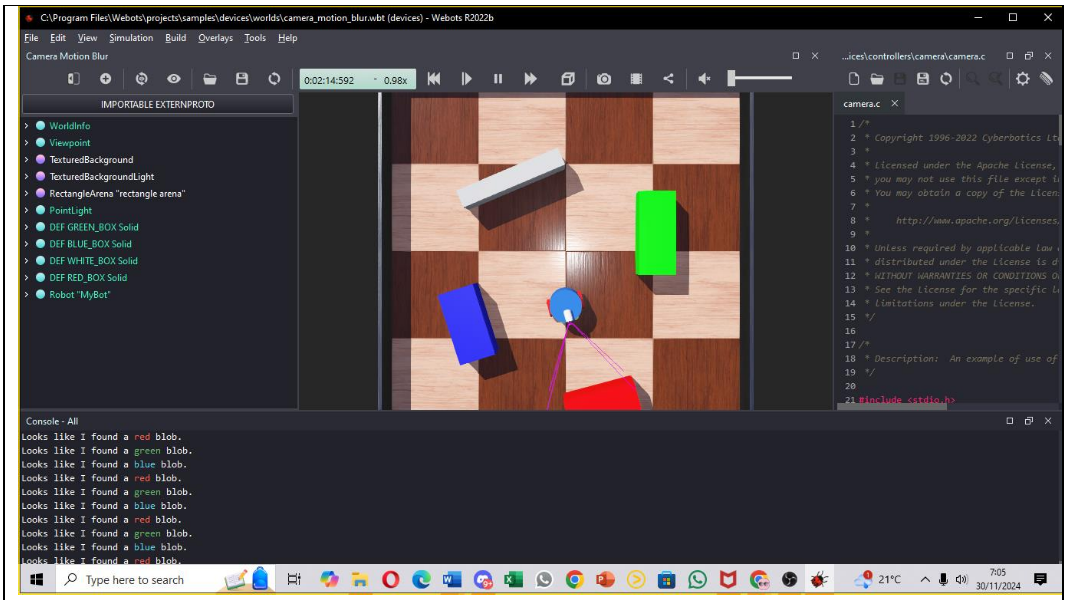
Output:



C:\Program Files\Webots\projects\samples\devices\worlds\camera_motion_blur.wbt (devices) - Webots R2022b

```
/*
 * Copyright 1996-2022 Cyberbotics Lt
 *
 * Licensed under the Apache License,
 * you may not use this file except i
 * You may obtain a copy of the Licen
 *
 *     http://www.apache.org/licenses,
 *
 * Unless required by applicable law
 * distributed under the License is d
 * WITHOUT WARRANTIES OR CONDITIONS O
 * See the License for the specific l
 * limitations under the License.
 */

/*
 * Description:  An example of use of
 */

#include <stdio.h>
```

```
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
```



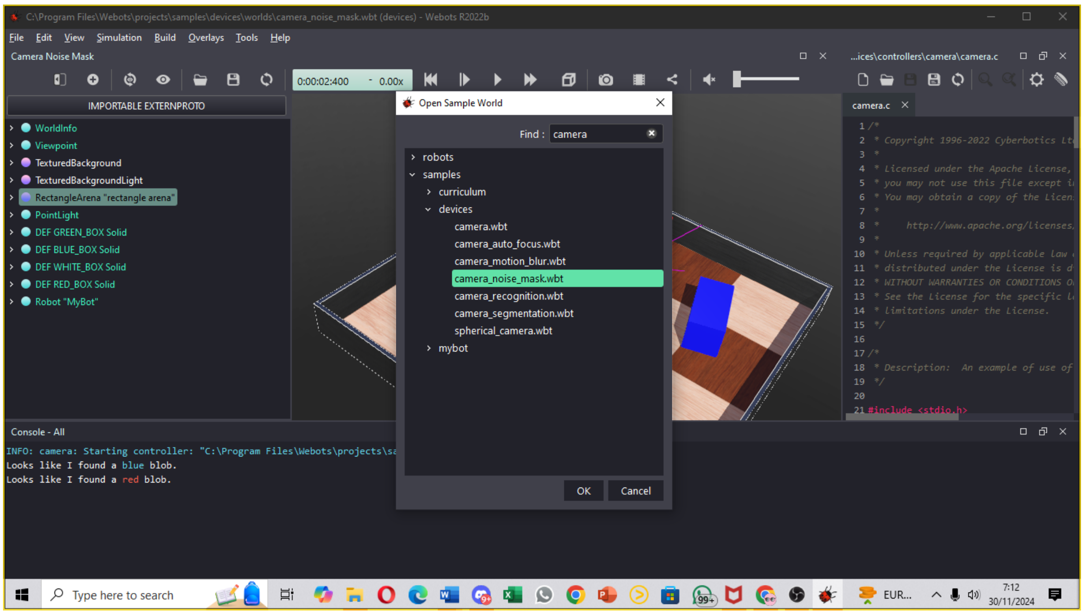
```
/*
 * Copyright 1996-2022 Cyberbotics Lt
 *
 * Licensed under the Apache License,
 * you may not use this file except i
 * You may obtain a copy of the Licen
 *
 *     http://www.apache.org/licenses,
 *
 * Unless required by applicable law
 * distributed under the License is d
 * WITHOUT WARRANTIES OR CONDITIONS O
 * See the License for the specific l
 * limitations under the License.
 */

/*
 * Description:  An example of use of
 */

#include <stdio.h>
```

```
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
```

Link Video Output:

https://drive.google.com/file/d/1uqPvnA67PCA3sHK43Ll4ltZeRuhT91xu/view?usp=sharing

d. Robot dengan kamera: deteksi blob berwarna dengan *noise mask*



```
/*
 * Copyright 1996-2022 Cyberbotics Lt
 *
 * Licensed under the Apache License,
 * you may not use this file except i
 * You may obtain a copy of the Licen
 *
 *     http://www.apache.org/licenses.
 *
 * Unless required by applicable law
 * distributed under the License is d
 * WITHOUT WARRANTIES OR CONDITIONS O
 * See the License for the specific l
 * limitations under the License.
 */

/*
 * Description:  An example of use of
 */

#include <stdio.h>
```

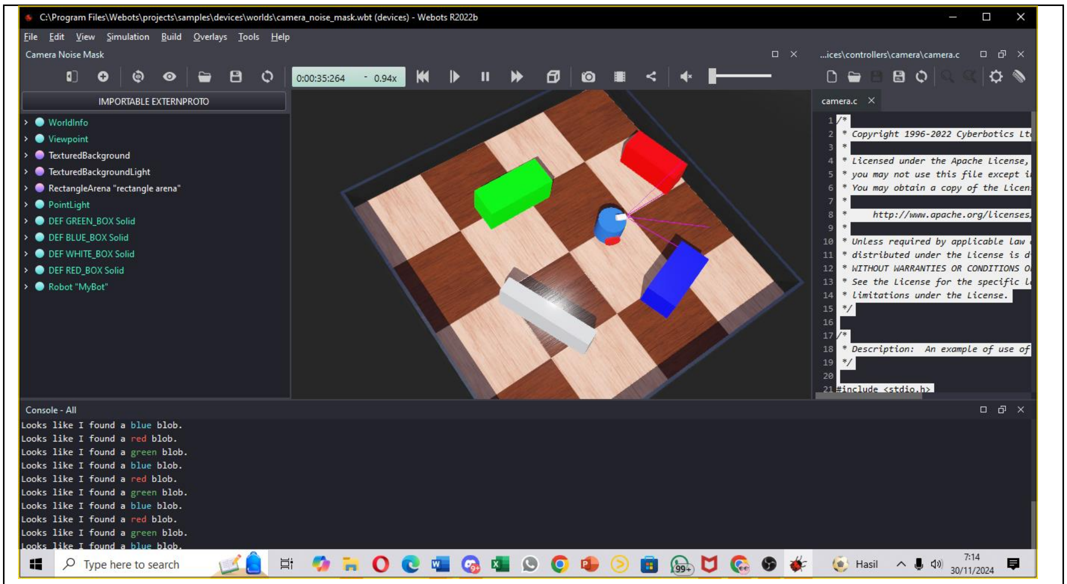
```
INFO: camera: Starting controller: "C:\Program Files\Webots\projects\sa
Looks like I found a blue blob.
Looks like I found a red blob.
```

Code:

```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the
"License");
 * you may not use this file except in compliance with the
License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in
writing, software
 * distributed under the License is distributed on an "AS
IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
 * See the License for the specific language governing
permissions and
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera device.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <webots/camera.h>
#include <webots/motor.h>
#include <webots/robot.h>
#include <webots/utils/system.h>

#define ANSI_COLOR_RED "\x1b[31m"
#define ANSI_COLOR_GREEN "\x1b[32m"
#define ANSI_COLOR_YELLOW "\x1b[33m"
#define ANSI_COLOR_BLUE "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN "\x1b[36m"
#define ANSI_COLOR_RESET "\x1b[0m"

#define SPEED 4
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int width, height;
  int pause_counter = 0;
  int left_speed, right_speed;
  int i, j;
  int red, blue, green;
  const char *color_names[3] = {"red", "green", "blue"};
  const char *ansi_colors[3] = {ANSI_COLOR_RED,
ANSI_COLOR_GREEN, ANSI_COLOR_BLUE};
  const char *filenames[3] = {"red_blob.png",
"green_blob.png", "blue_blob.png"};
  enum BLOB_TYPE current_blob;

  wb_robot_init();

  const int time_step = wb_robot_get_basic_time_step();
  /* Get the camera device, enable it, and store its width
and height */
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, time_step);
  width = wb_camera_get_width(camera);
  height = wb_camera_get_height(camera);

  /* get a handler to the motors and set target position
to infinity (speed control). */
  left_motor = wb_robot_get_device("left wheel motor");
  right_motor = wb_robot_get_device("right wheel motor");
  wb_motor_set_position(left_motor, INFINITY);
  wb_motor_set_position(right_motor, INFINITY);
  wb_motor_set_velocity(left_motor, 0.0);
  wb_motor_set_velocity(right_motor, 0.0);

  /* Main loop */
  while (wb_robot_step(time_step) != -1) {
    /* Get the new camera values */
    const unsigned char *image =
wb_camera_get_image(camera);

    /* Decrement the pause_counter */
    if (pause_counter > 0)
      pause_counter--;

    /*
     * Case 1
     * A blob was found recently
     * The robot waits in front of it until pause_counter
     * is decremented enough
     */
    if (pause_counter > 640 / time_step) {
      left_speed = 0;
      right_speed = 0;
    }
    /*
     * Case 2
     * A blob was found quite recently
     * The robot begins to turn but don't analyse the
image for a while,
     * otherwise the same blob would be found again
     */
    else if (pause_counter > 0) {
      left_speed = -SPEED;
      right_speed = SPEED;
    }
    /*
     * Case 3
     * The robot turns and analyse the camera image in
order
     * to find a new blob
     */
    else if (!image) {  // image may be NULL if
Robot.synchronization is FALSE
```

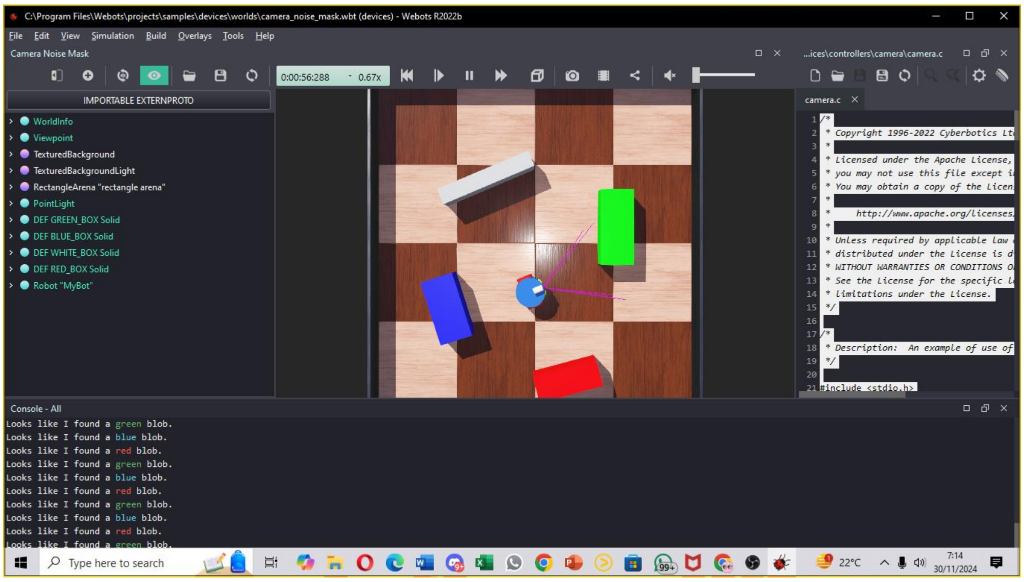
```
      left_speed = 0;
      right_speed = 0;
    } else {  // pause_counter == 0
      /* Reset the sums */
      red = 0;
      green = 0;
      blue = 0;

      /*
       * Here we analyse the image from the camera. The
goal is to detect a
       * blob (a spot of color) of a defined color in the
middle of our
       * screen.
       * In order to achieve that we simply parse the
image pixels of the
       * center of the image, and sum the color components
individually
       */
      for (i = width / 3; i < 2 * width / 3; i++) {
        for (j = height / 2; j < 3 * height / 4; j++) {
          red += wb_camera_image_get_red(image, width, i,
j);
          blue += wb_camera_image_get_blue(image, width,
i, j);
          green += wb_camera_image_get_green(image, width,
i, j);
        }
      }

      /*
       * If a component is much more represented than the
other ones,
       * a blob is detected
       */
      if ((red > 3 * green) && (red > 3 * blue))
        current_blob = RED;
      else if ((green > 3 * red) && (green > 3 * blue))
        current_blob = GREEN;
      else if ((blue > 3 * red) && (blue > 3 * green))
        current_blob = BLUE;
      else
        current_blob = NONE;

      /*
       * Case 3a
       * No blob is detected
       * the robot continues to turn
       */
      if (current_blob == NONE) {
        left_speed = -SPEED;
        right_speed = SPEED;
      }
      /*
       * Case 3b
       * A blob is detected
       * the robot stops, stores the image, and changes
its state
       */
      else {
        left_speed = 0;
        right_speed = 0;
        printf("Looks like I found a %s%s%s blob.\n",
ansi_colors[current_blob], color_names[current_blob],
ANSI_COLOR_RESET);
        // compute the file path in the user directory
        char *filepath;
#ifdef _WIN32
        const char *user_directory =
wbu_system_short_path(wbu_system_getenv("USERPROFILE"));
        filepath = (char *)malloc(strlen(user_directory) +
16);
        strcpy(filepath, user_directory);
        strcat(filepath, "\\");
#else
        const char *user_directory =
wbu_system_getenv("HOME");
        filepath = (char *)malloc(strlen(user_directory) +
16);
        strcpy(filepath, user_directory);
        strcat(filepath, "/");
#endif
        strcat(filepath, filenames[current_blob]);
        wb_camera_save_image(camera, filepath, 100);
        free(filepath);
        pause_counter = 1280 / time_step;
      }
    }

    /* Set the motor speeds. */
    wb_motor_set_velocity(left_motor, left_speed);
    wb_motor_set_velocity(right_motor, right_speed);
  }

  wb_robot_cleanup();

  return 0;
}
```

Output:



```
/*
 * Copyright 1996-2022 Cyberbotics Lt
 *
 * Licensed under the Apache License,
 * you may not use this file except i
 * You may obtain a copy of the Licen
 *
 *     http://www.apache.org/licenses
 *
 * Unless required by applicable law
 * distributed under the License is d
 * WITHOUT WARRANTIES OR CONDITIONS O
 * See the License for the specific l
 * limitations under the License.
 */

/*
 * Description:  An example of use of
 */

#include <stdio.h>
```

```
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
```



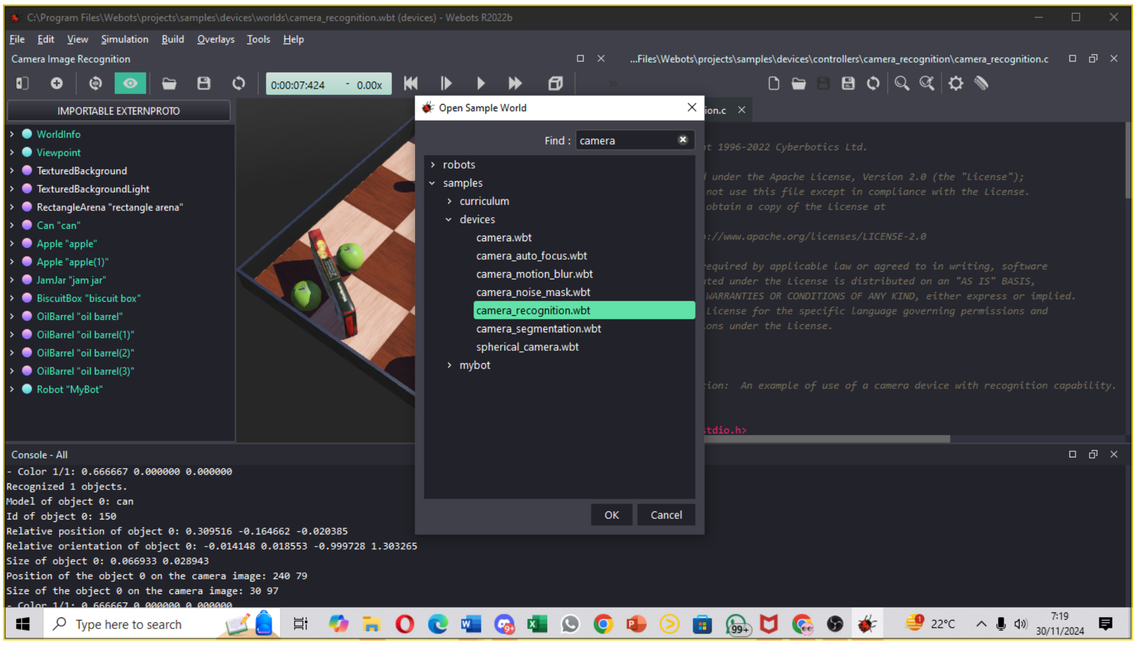
```
/*
 * Copyright 1996-2022 Cyberbotics Lt
 *
 * Licensed under the Apache License,
 * you may not use this file except i
 * You may obtain a copy of the Licen
 *
 *     http://www.apache.org/licenses
 *
 * Unless required by applicable law
 * distributed under the License is d
 * WITHOUT WARRANTIES OR CONDITIONS O
 * See the License for the specific l
 * limitations under the License.
 */

/*
 * Description:  An example of use of
 */

#include <stdio.h>
```

```
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
```

Link Video Output:

https://drive.google.com/file/d/1SitZg1Zscpw-gTI5bPPFpkpVoN6ceNPm/view?usp=sharing

e. Deteksi objek dengan kamera dan pengenalan objek pada robot (*recognition*)



```
- Color 1/1: 0.666667 0.000000 0.000000
Recognized 1 objects.
Model of object 0: can
Id of object 0: 150
Relative position of object 0: 0.309516 -0.164662 -0.020385
Relative orientation of object 0: -0.014148 0.018553 -0.999728 1.303265
Size of object 0: 0.066933 0.028943
Position of the object 0 on the camera image: 240 79
Size of the object 0 on the camera image: 30 97
- Color 1/1: 0.666667 0.000000 0.000000
```

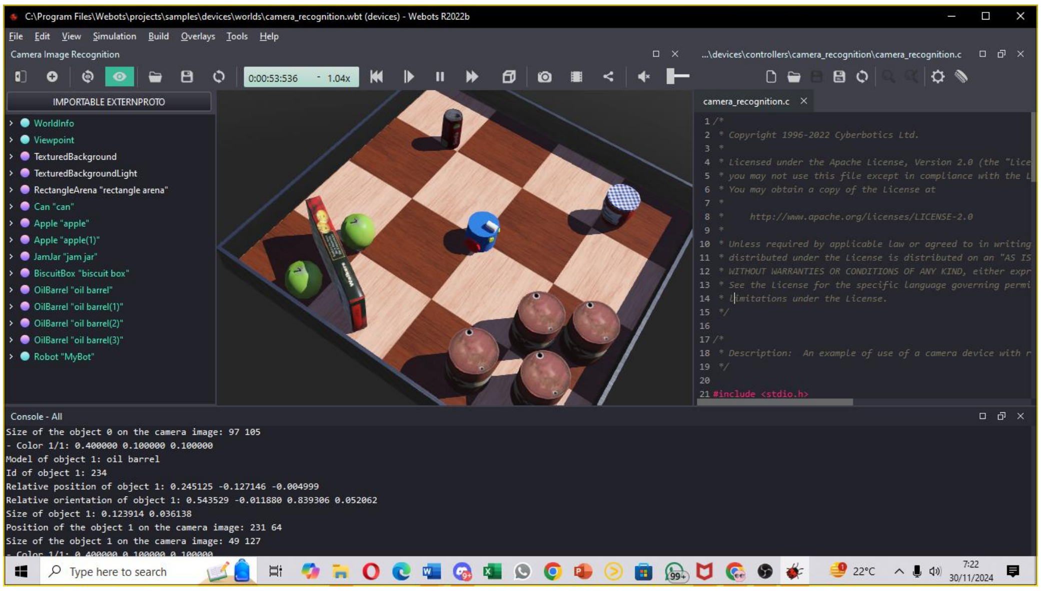
Code:

```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0
(the "License");
 * you may not use this file except in compliance
with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in
writing, software
 * distributed under the License is distributed on an
"AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
either express or implied.
 * See the License for the specific language
governing permissions and
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera device
with recognition capability.
 */

#include <stdio.h>
#include <webots/camera.h>
#include <webots/camera_recognition_object.h>
#include <webots/motor.h>
#include <webots/robot.h>
```

```
#define SPEED 1.5
#define TIME_STEP 64

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int i, j;

  wb_robot_init();

  /* Get the camera device, enable it and the
recognition */
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, TIME_STEP);
  wb_camera_recognition_enable(camera, TIME_STEP);

  /* get a handler to the motors and set target
position to infinity (speed control). */
  left_motor = wb_robot_get_device("left wheel
motor");
  right_motor = wb_robot_get_device("right wheel
motor");
  wb_motor_set_position(left_motor, INFINITY);
  wb_motor_set_position(right_motor, INFINITY);
  wb_motor_set_velocity(left_motor, 0.0);
  wb_motor_set_velocity(right_motor, 0.0);

  /* Set the motors speed */
  wb_motor_set_velocity(left_motor, -SPEED);
  wb_motor_set_velocity(right_motor, SPEED);

  /* Main loop */
  while (wb_robot_step(TIME_STEP) != -1) {
    /* Get current number of object recognized */
    int number_of_objects =
wb_camera_recognition_get_number_of_objects(camera);
    printf("\nRecognized %d objects.\n",
number_of_objects);

    /* Get and display all the objects information */
    const WbCameraRecognitionObject *objects =
wb_camera_recognition_get_objects(camera);
    for (i = 0; i < number_of_objects; ++i) {
      printf("Model of object %d: %s\n", i,
objects[i].model);
      printf("Id of object %d: %d\n", i,
objects[i].id);
      printf("Relative position of object %d: %lf %lf
%lf\n", i, objects[i].position[0],
objects[i].position[1],
             objects[i].position[2]);
      printf("Relative orientation of object %d: %lf
%lf %lf %lf\n", i, objects[i].orientation[0],
objects[i].orientation[1],
             objects[i].orientation[2],
objects[i].orientation[3]);
```

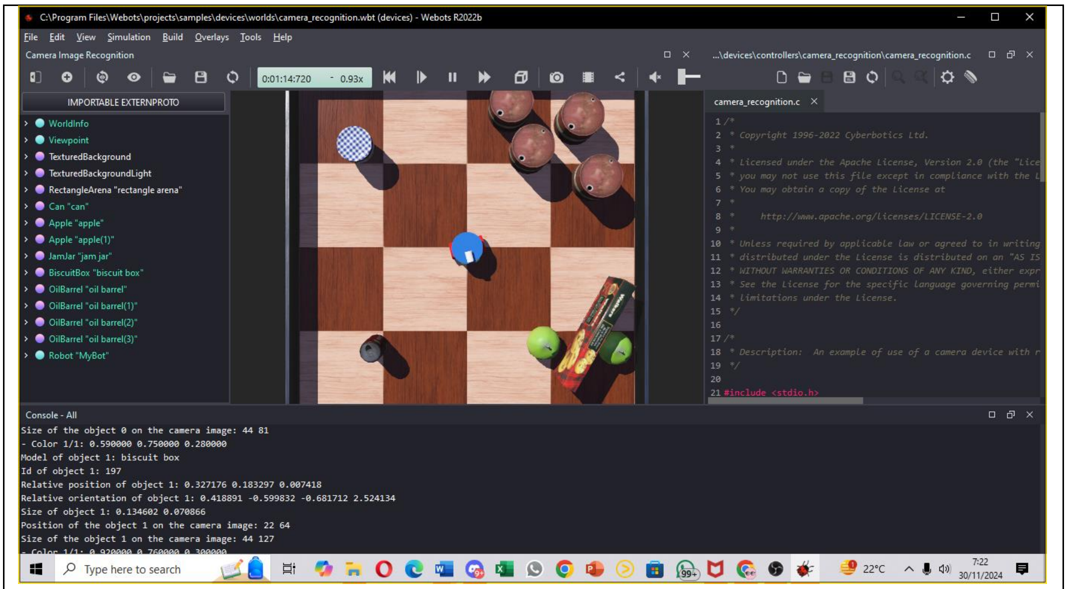
```
          printf("Size of object %d: %lf %lf\n", i,
      objects[i].size[0], objects[i].size[1]);
          printf("Position of the object %d on the camera
      image: %d %d\n", i, objects[i].position_on_image[0],
                 objects[i].position_on_image[1]);
          printf("Size of the object %d on the camera
      image: %d %d\n", i, objects[i].size_on_image[0],
      objects[i].size_on_image[1]);
          for (j = 0; j < objects[i].number_of_colors;
      ++j)
            printf("- Color %d/%d: %lf %lf %lf\n", j + 1,
      objects[i].number_of_colors, objects[i].colors[3 *
      j],
                   objects[i].colors[3 * j + 1],
      objects[i].colors[3 * j + 2]);
          }
        }

        wb_robot_cleanup();

        return 0;
}
```

Output:


C:\Program Files\Webots\projects\samples\devices\worlds\camera_recognition.wbt (devices) - Webots R2022b
File Edit View Simulation Build Overlays Tools Help
Camera Image Recognition
0:00:53:536
1.04x
IMPORTABLE EXTERNPROTO
WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
Can "can"
Apple "apple"
Apple "apple(1)"
JamJar "jam jar"
BiscuitBox "biscuit box"
OilBarrel "oil barrel"
OilBarrel "oil barrel(1)"
OilBarrel "oil barrel(2)"
OilBarrel "oil barrel(3)"
Robot "MyBot"
...\devices\controllers\camera_recognition\camera_recognition.c
camera_recognition.c
Console - All
Type here to search
22°C
7:22
30/11/2024

```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "Lice
 * you may not use this file except in compliance with the L
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing
 * distributed under the License is distributed on an "AS IS
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expr
 * See the License for the specific language governing permi
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera device with r
 */

#include <stdio.h>
```

```
Size of the object 0 on the camera image: 97 105
- Color 1/1: 0.400000 0.100000 0.100000
Model of object 1: oil barrel
Id of object 1: 234
Relative position of object 1: 0.245125 -0.127146 -0.004999
Relative orientation of object 1: 0.543529 -0.011880 0.839306 0.052062
Size of object 1: 0.123914 0.036138
Position of the object 1 on the camera image: 231 64
Size of the object 1 on the camera image: 49 127
- Color 1/1: 0.400000 0.100000 0.100000
```


C:\Program Files\Webots\projects\samples\devices\worlds\camera_recognition.wbt (devices) - Webots R2022b
File Edit View Simulation Build Overlays Tools Help
Camera Image Recognition
0:01:14:720 0.93x
IMPORTABLE EXTERNPROTO
WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
Can "can"
Apple "apple"
Apple "apple(1)"
JamJar "jam jar"
BiscuitBox "biscuit box"
OilBarrel "oil barrel"
OilBarrel "oil barrel(1)"
OilBarrel "oil barrel(2)"
OilBarrel "oil barrel(3)"
Robot "MyBot"
...\devices\controllers\camera_recognition\camera_recognition.c
camera_recognition.c
Console - All
Type here to search
22°C
7:22
30/11/2024

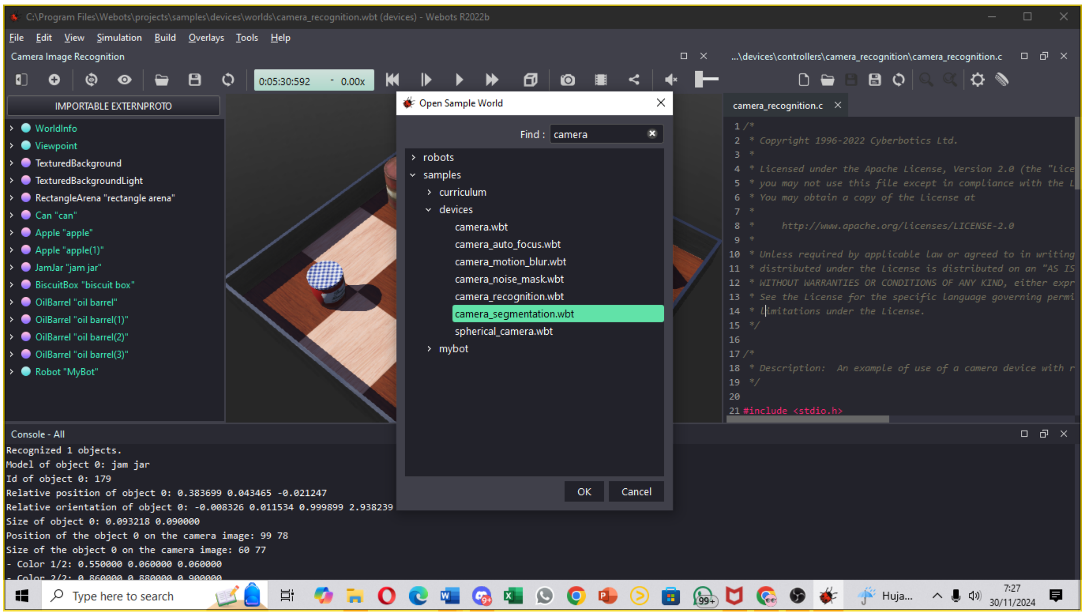
```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "Lice
 * you may not use this file except in compliance with the L
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing
 * distributed under the License is distributed on an "AS IS
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expr
 * See the License for the specific language governing permi
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera device with r
 */

#include <stdio.h>
```

```
Size of the object 0 on the camera image: 44 81
- Color 1/1: 0.590000 0.750000 0.280000
Model of object 1: biscuit box
Id of object 1: 197
Relative position of object 1: 0.327176 0.183297 0.007418
Relative orientation of object 1: 0.418891 -0.599832 -0.681712 2.524134
Size of object 1: 0.134602 0.070866
Position of the object 1 on the camera image: 22 64
Size of the object 1 on the camera image: 44 127
- Color 1/1: 0.920000 0.760000 0.300000
```

```
- Color 1/2: 0.550000 0.060000 0.060000
- Color 2/2: 0.860000 0.880000 0.900000
Recognized 1 objects.
Model of object 0: jam jar
Id of object 0: 179
Relative position of object 0: 0.369144 -0.118401 -0.021659
Relative orientation of object 0: -0.010775 0.010119 0.999891 2.555273
Size of object 0: 0.093218 0.090000
Position of the object 0 on the camera image: 204 79
Size of the object 0 on the camera image: 73 81
- Color 1/2: 0.550000 0.060000 0.060000
- Color 2/2: 0.860000 0.880000 0.900000
Recognized 1 objects.
Model of object 0: jam jar
Id of object 0: 179
Relative position of object 0: 0.362988 -0.137958 -0.021834
Relative orientation of object 0: -0.011098 0.009932 0.999889 2.507249
Size of object 0: 0.093218 0.090000
Position of the object 0 on the camera image: 217 79
Size of the object 0 on the camera image: 77 82
- Color 1/2: 0.550000 0.060000 0.060000
- Color 2/2: 0.860000 0.880000 0.900000
Recognized 1 objects.
Model of object 0: jam jar
Id of object 0: 179
Relative position of object 0: 0.355901 -0.155603 -0.022034
Relative orientation of object 0: -0.011427 0.009743 0.999887 2.459231
Size of object 0: 0.093218 0.086816
Position of the object 0 on the camera image: 222 79
Size of the object 0 on the camera image: 66 84
- Color 1/2: 0.550000 0.060000 0.060000
- Color 2/2: 0.860000 0.880000 0.900000
Recognized 1 objects.
Model of object 0: jam jar
```

Id of object 0: 179 Relative position of object 0: 0.347900 -0.164303 -0.022261 Relative orientation of object 0: -0.011761 0.009549 0.999885 2.411216 Size of object 0: 0.093218 0.066472
Link Video Output: https://drive.google.com/file/d/12k9NmSy3Pkc4Ua7UWgdhG5z80g5iPE_l/view?usp=sharing

f. Implementasi segmentasi kamera pada robot menggunakan webots



```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "Lice
 * you may not use this file except in compliance with the L
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing
 * distributed under the License is distributed on an "AS IS
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expr
 * See the License for the specific language governing permi
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera device with r
 */

#include <stdio.h>
```

```
Recognized 1 objects.
Model of object 0: jam jar
Id of object 0: 179
Relative position of object 0: 0.383699 0.043465 -0.021247
Relative orientation of object 0: -0.008326 0.011534 0.999899 2.938239
Size of object 0: 0.093218 0.090000
Position of the object 0 on the camera image: 99 78
Size of the object 0 on the camera image: 60 77
- Color 1/2: 0.550000 0.060000 0.060000
- Color 2/2: 0.860000 0.880000 0.900000
```

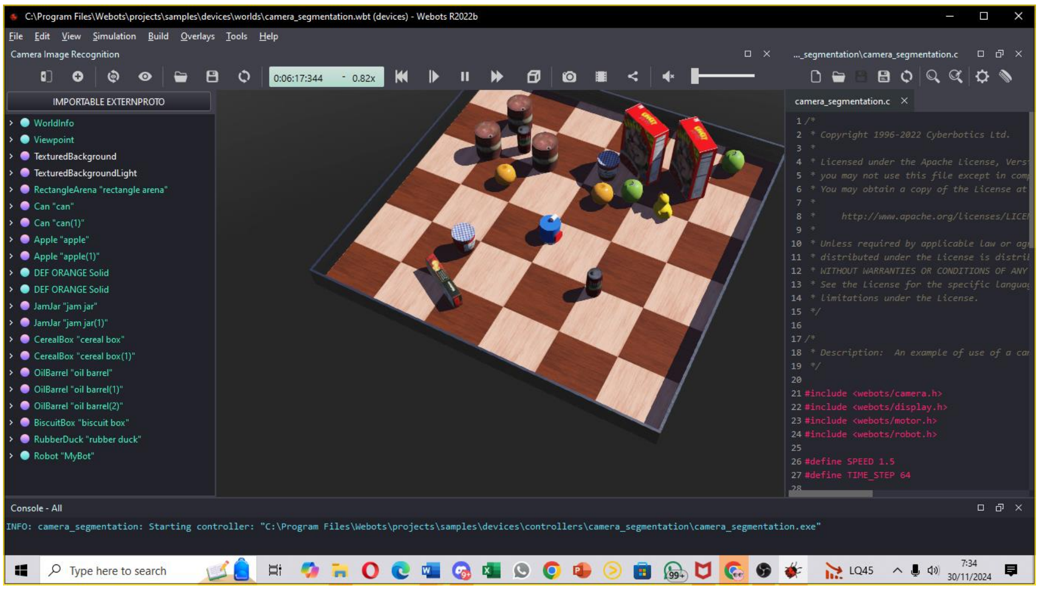
Code:

```
        /*
         * Copyright 1996-2022 Cyberbotics Ltd.
         *
         * Licensed under the Apache License, Version 2.0
        (the "License");
         * you may not use this file except in compliance
        with the License.
         * You may obtain a copy of the License at
         *
         *     http://www.apache.org/licenses/LICENSE-2.0
         *
         * Unless required by applicable law or agreed to in
        writing, software
         * distributed under the License is distributed on an
        "AS IS" BASIS,
         * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
        either express or implied.
         * See the License for the specific language
        governing permissions and
         * limitations under the License.
         */
```

```
/*
 * Description:  An example of use of a camera device
with recognition segmentation capability.
 */

#include <webots/camera.h>
#include <webots/display.h>
#include <webots/motor.h>
#include <webots/robot.h>

#define SPEED 1.5
#define TIME_STEP 64

int main() {
  WbDeviceTag camera, display, left_motor,
right_motor;
  WbImageRef segmented_image;

  wb_robot_init();

  /* Get the camera device, enable the camera, the
recognition and the segmentation functionalities */
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, TIME_STEP);
  wb_camera_recognition_enable(camera, TIME_STEP);
  wb_camera_recognition_enable_segmentation(camera);
  const int width = wb_camera_get_width(camera);
  const int height = wb_camera_get_height(camera);

  /* Get the display device */
  display = wb_robot_get_device("segmented image
display");

  /* Get a handler to the motors and set target
position to infinity (speed control). */
  left_motor = wb_robot_get_device("left wheel
motor");
  right_motor = wb_robot_get_device("right wheel
motor");
  wb_motor_set_position(left_motor, INFINITY);
  wb_motor_set_position(right_motor, INFINITY);
  wb_motor_set_velocity(left_motor, 0.0);
  wb_motor_set_velocity(right_motor, 0.0);

  /* Set the motors speed */
  wb_motor_set_velocity(left_motor, -SPEED);
  wb_motor_set_velocity(right_motor, SPEED);

  /* Main loop */
  while (wb_robot_step(TIME_STEP) != -1) {
    if
(wb_camera_recognition_is_segmentation_enabled(camera
) &&
wb_camera_recognition_get_sampling_period(camera) >
0) {
```

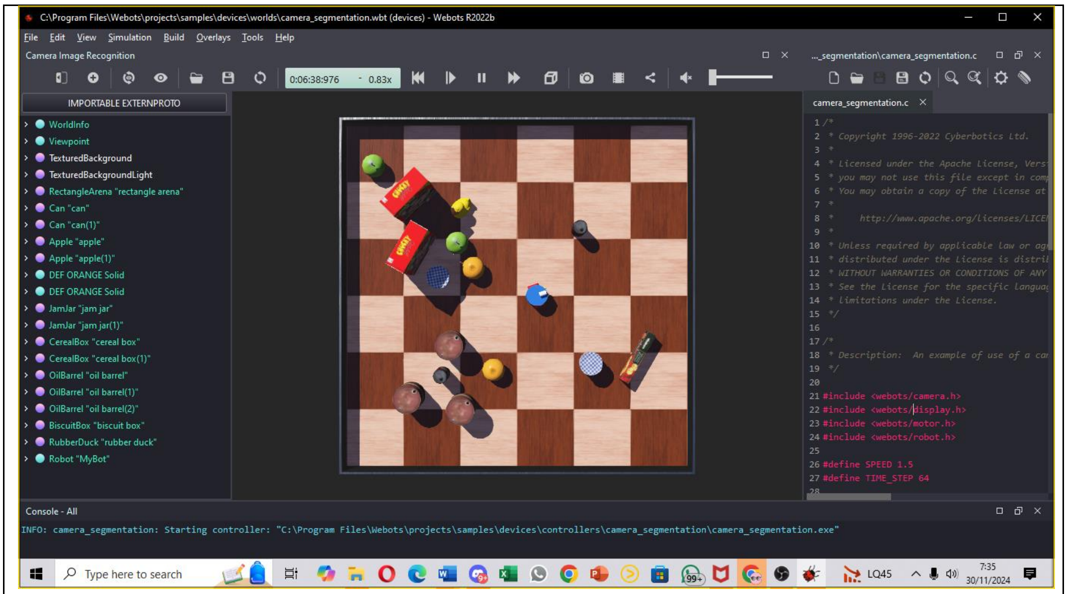
```
            /* Get the segmented image and display it in
      the Display */
            const unsigned char *data =
      wb_camera_recognition_get_segmentation_image(camera);
            if (data) {
              segmented_image =
      wb_display_image_new(display, width, height, data,
      WB_IMAGE_BGRA);
              wb_display_image_paste(display,
      segmented_image, 0, 0, false);
              wb_display_image_delete(display,
      segmented_image);
            }
          }
        }

        wb_robot_cleanup();

        return 0;
}
```

Output:



```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Vers
 * you may not use this file except in com
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICE
 *
 * Unless required by applicable law or ag
 * distributed under the License is distri
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * See the License for the specific langua
 * limitations under the License.
 */

/*
 * Description:  An example of use of a ca
 */

#include <webots/camera.h>
#include <webots/display.h>
#include <webots/motor.h>
#include <webots/robot.h>

#define SPEED 1.5
#define TIME_STEP 64
```

```
INFO: camera_segmentation: Starting controller: "C:\Program Files\Webots\projects\samples\devices\controllers\camera_segmentation\camera_segmentation.exe"
```



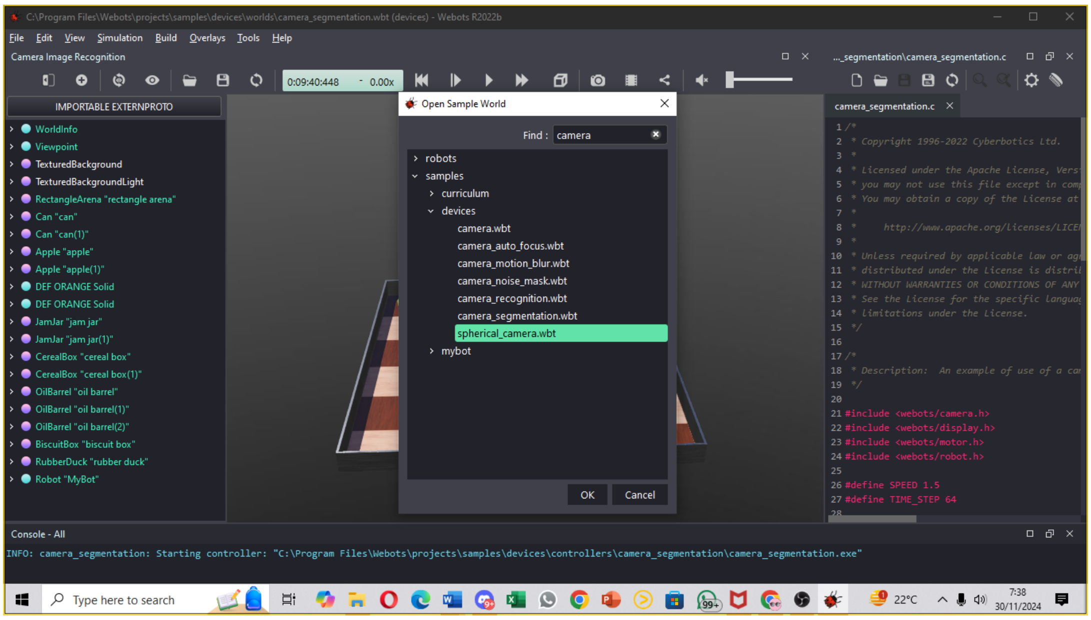
```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Vers
 * you may not use this file except in comp
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICEN
 *
 * Unless required by applicable law or agr
 * distributed under the License is distrib
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * See the License for the specific languag
 * limitations under the License.
 */

/*
 * Description:  An example of use of a car
 */

#include <webots/camera.h>
#include <webots/display.h>
#include <webots/motor.h>
#include <webots/robot.h>

#define SPEED 1.5
#define TIME_STEP 64
```

```
INFO: camera_segmentation: Starting controller: "C:\Program Files\Webots\projects\samples\devices\controllers\camera_segmentation\camera_segmentation.exe"
```

Link Video Output:

https://drive.google.com/file/d/1fi7SOaQCgiDBi-BpOBEH2YBk5i9MdRUC/view?usp=sharing

g. Implementasi penggunaan kamera bola pada robot menggunakan webots



```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Vers
 * you may not use this file except in comp
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICEN
 *
 * Unless required by applicable law or agr
 * distributed under the License is distri
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * See the License for the specific languag
 * limitations under the License.
 */

/*
 * Description:  An example of use of a car
 */

#include <webots/camera.h>
#include <webots/display.h>
#include <webots/motor.h>
#include <webots/robot.h>

#define SPEED 1.5
#define TIME_STEP 64
```

```
INFO: camera_segmentation: Starting controller: "C:\Program Files\Webots\projects\samples\devices\controllers\camera_segmentation\camera_segmentation.exe"
```

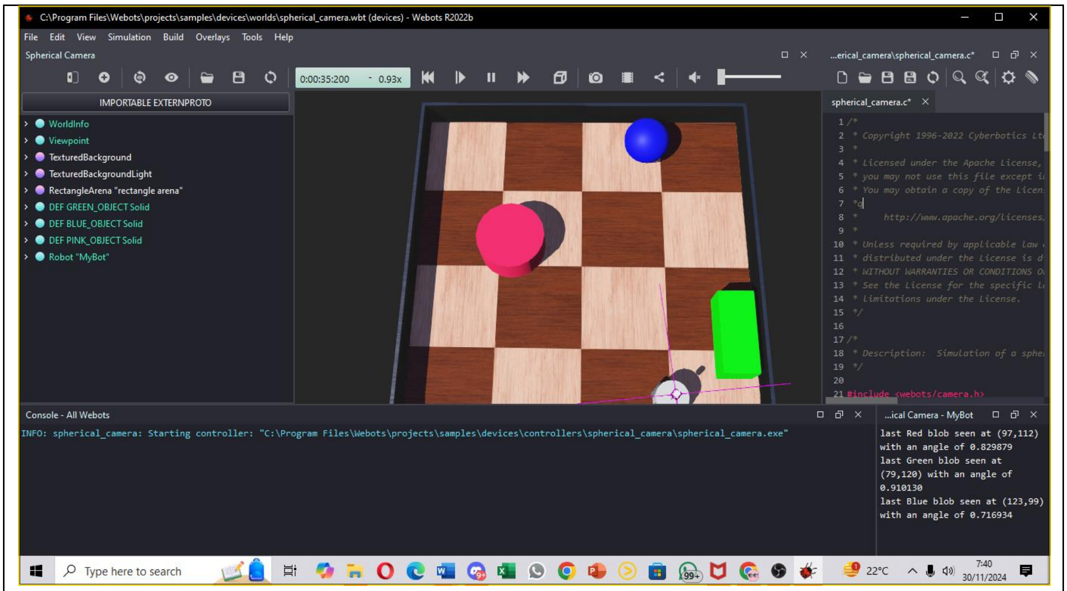
Code:

```
/*
 * Copyright 1996-2022 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the
"License");
 * you may not use this file except in compliance with the
License.
```

```
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in
writing, software
 * distributed under the License is distributed on an "AS
IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
 * See the License for the specific language governing
permissions and
 * limitations under the License.
 */

/*
 * Description:  Simulation of a spherical camera
 */

#include <webots/camera.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>
#include <webots/robot.h>
#include <webots/utils/ansi_codes.h>

#include <math.h>
#include <stdio.h>

#define TIME_STEP 64
#define THRESHOLD 200

#define RED 0
#define GREEN 1
#define BLUE 2

#define LEFT 0
#define RIGHT 1

#define X 0
#define Y 1

double coord2D_to_angle(double x, double y) {
  if (x > 0.0 && y >= 0.0)
    return atan(y / x);
  else if (x > 0.0 && y < 0.0)
    return atan(y / x) + 2.0 * M_PI;
  else if (x < 0.0)
    return atan(y / x) + M_PI;
  else if (x == 0.0 && y > 0.0)
    return M_PI_2;
  else if (x == 0.0 && y < 0.0)
    return 3.0 * M_PI_2;
  else /* (x == 0.0 && y == 0.0) */
    return 0.0;
}
```

```
int main(int argc, char **argv) {
  // iterator used to parse loops
  int i, k;

  // init Webots stuff
  wb_robot_init();

  // init camera
  WbDeviceTag camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, 2 * TIME_STEP);
  int width = wb_camera_get_width(camera);
  int height = wb_camera_get_height(camera);
  int color_index[3][2] = {{0, 0}, {0, 0}, {0, 0}};
  int x, y, r, g, b;

  // init distance sensors
  WbDeviceTag us[2];
  double us_values[2];
  double coefficients[2][2] = {{6.0, -3.0}, {-5.0, 4.0}};
  us[LEFT] = wb_robot_get_device("us0");
  us[RIGHT] = wb_robot_get_device("us1");
  for (i = 0; i < 2; i++)
    wb_distance_sensor_enable(us[i], TIME_STEP);

  // get a handler to the motors and set target position
to infinity (speed control)
  WbDeviceTag left_motor = wb_robot_get_device("left wheel
motor");
  WbDeviceTag right_motor = wb_robot_get_device("right
wheel motor");
  wb_motor_set_position(left_motor, INFINITY);
  wb_motor_set_position(right_motor, INFINITY);
  wb_motor_set_velocity(left_motor, 0.0);
  wb_motor_set_velocity(right_motor, 0.0);

  // init speed values
  double speed[2];

  while (wb_robot_step(TIME_STEP) != -1) {
    // read sensors
    const unsigned char *image =
wb_camera_get_image(camera);
    for (i = 0; i < 2; i++)
      us_values[i] = wb_distance_sensor_get_value(us[i]);

    // compute speed
    for (i = 0; i < 2; i++) {
      speed[i] = 0.0;
      for (k = 0; k < 2; k++)
        speed[i] += us_values[k] * coefficients[i][k];
    }

    // compute blob direction
    for (y = 0; y < height; y++) {
      for (x = 0; x < width; x++) {
        r = wb_camera_image_get_red(image, width, x, y);
        g = wb_camera_image_get_green(image, width, x, y);
        b = wb_camera_image_get_blue(image, width, x, y);
        if (r > THRESHOLD && g < THRESHOLD && b <
THRESHOLD) {
          color_index[RED][X] = x;
          color_index[RED][Y] = y;
        } else if (r < THRESHOLD && g > THRESHOLD && b <
THRESHOLD) {
          color_index[GREEN][X] = x;
          color_index[GREEN][Y] = y;
        } else if (r < THRESHOLD && g < THRESHOLD && b >
THRESHOLD) {
          color_index[BLUE][X] = x;
          color_index[BLUE][Y] = y;
        }
      }
    }

    // print results
    ANSI_CLEAR_CONSOLE();
    for (i = 0; i < 3; i++)
      // clang-format off
      // clang-format 11.0.0 is not compatible with
previous versions with respect to nested conditional
operators
      printf("last %s blob seen at (%d,%d) with an angle
of %f\n",
             (i == GREEN) ? "Green" :
             (i == RED)   ? "Red" :
                            "Blue",
             color_index[i][X], color_index[i][Y],
             coord2D_to_angle((double)(color_index[i][X] +
width / 2), (double)(color_index[i][Y] + height / 2)));
    // clang-format on

    // set actuators
    wb_motor_set_velocity(left_motor, 3.0 + speed[LEFT]);
    wb_motor_set_velocity(right_motor, 3.0 +
speed[RIGHT]);
  }

  wb_robot_cleanup();

  return 0;
}
```

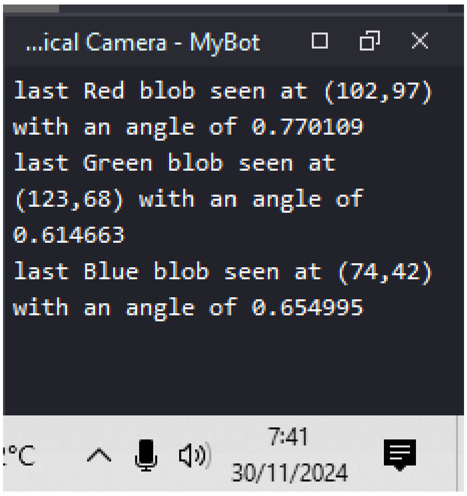
Output:



```
/*
 * Copyright 1996-2022 Cyberbotics Lt
 *
 * Licensed under the Apache License,
 * you may not use this file except i
 * You may obtain a copy of the Licen
 *d
 *     http://www.apache.org/licenses/
 *
 * Unless required by applicable law 
 * distributed under the License is d
 * WITHOUT WARRANTIES OR CONDITIONS O
 * See the License for the specific l
 * limitations under the License.
 */

/*
 * Description:  Simulation of a sphe
 */

#include <webots/camera.h>
```

```
INFO: spherical_camera: Starting controller: "C:\Program Files\Webots\projects\samples\devices\controllers\spherical_camera\spherical_camera.exe"
```

```
last Red blob seen at (97,112)
with an angle of 0.829879
last Green blob seen at
(79,120) with an angle of
0.910130
last Blue blob seen at (123,99)
with an angle of 0.716934
```



```
last Red blob seen at (102,97)
with an angle of 0.770109
last Green blob seen at
(123,68) with an angle of
0.614663
last Blue blob seen at (74,42)
with an angle of 0.654995
```

Link Video Output:

https://drive.google.com/file/d/1ZWZNIkQDDg8AYqyB3re1V8-C0VRhx-Ws/view?usp=sharing