

# Evidence of Use of Version Control

## Group 23

Lachlan Chow z5164192

James Davies z5162711

Author	Commit	Message	Date
James Davies	eaabfc8	wheel_style_improvements	2 minutes ago
Lachlan Chow	af21b4b	main_style_improvements	8 minutes ago
James Davies	6682a11	vehicle_cpp_style_improvements	13 minutes ago
James Davies	34d15d1	vehicle_h_style_improvements	40 minutes ago
James Davies	65a65f2	trapezoid_style_improvements	47 minutes ago
James Davies	af7929f	Cylinder_style_improvements	58 minutes ago
Lachlan Chow	750b338	RectangularPrism_style_improvements	an hour ago
James Davies	4842b5b	TriangularPrism_style_improvements	an hour ago
Lachlan Chow	6504f65	modify_variable_names	2 days ago
Lachlan Chow	6821c72	incorporate_xbox_controller	2 days ago
Lachlan Chow	84c2481	personal_vehicle_draw_functions_and_get_server_shape_function	2 days ago
Lachlan Chow	5a0280a	add_rotation_and_steering_booleans	2 days ago
Lachlan Chow	fe6a825	fix_wheel_display_shape	2 days ago
Lachlan Chow	776c094	Instantiate_server_vehicles	3 days ago
Lachlan Chow	69a47f2	Working-trapezoidprism	2018-09-04
Lachlan Chow	44c7027	Merged master into test	2018-09-04
Lachlan Chow	8c7d5ee	Basic vehicle class	2018-09-04
James Davies	f76f56d	finished-trapezoidprism	2018-09-04
unknown	8c13d73	Working-triangularprism-and-rectangularprism	2018-09-04
unknown	977557f	Working-cylinder-files	2018-09-04
unknown	68a2d99	finished_assignment	2018-09-04
James Davies	80c79ef	contributors.txt edited online with Bitbucket	2018-08-31
James Davies	x18d8ff	README.md edited online with Bitbucket	2018-08-31
Lachlan Chow	2d4d4f9	contributors.txt created online with Bitbucket	2018-08-31
Lachlan Chow	x388b79	README.md edited online with Bitbucket	2018-08-31
Lachlan Chow	8c2d33d	Initial commit	2018-08-30

Some examples of the raw commits

From eaabfc86a3cd790dc1ad62f2cc3bdff2495d6c64 Mon Sep 17 00:00:00 2001

From: james davies <james.davies444@gmail.com>

Date: Sat, 15 Sep 2018 08:33:09 +0000

Subject: [PATCH] wheel\_style\_improvements

---

wheel\_stylish | 140

+++++

1 file changed, 140 insertions(+)

create mode 100644 wheel\_stylish

diff --git a/wheel\_stylish b/wheel\_stylish

```
new file mode 100644
index 0000000..2b60200
--- /dev/null
+++ b/wheel_stylish
@@ -0,0 +1,140 @@
+//Lachlan Chow & James Davies
+//Assignment 2 MTRN 2500
+
+#include "RectangularPrism.h"
+#include "TrapezoidalPrism.h"
+#include "TriangularPrism.h"
+#include "Cylinder.h"
+#include "Wheel.h"
+#include "Vehicle1.h"
+#include "Vehicle.hpp"
+#include "Shape.hpp"
+#include <iostream>
+#include <cstdlib>
+#include <cstdio>
+#include <cstring>
+#include <sstream>
+#include <map>
+#include <math.h>
+
+#ifdef APPLE
+#include <OpenGL/gl.h>
+#include <OpenGL/glu.h>
+#include <GLUT/glut.h>
+#include <unistd.h>
+#include <sys/time.h>
+#elif defined(WIN32)
+#include <Windows.h>
+#include <tchar.h>
+#include <GL/gl.h>
+#include <GL/glu.h>
+#include <GL/glut.h>
+#else
+#include <GL/gl.h>
+#include <GL/glu.h>
+#include <GL/glut.h>
+#include <unistd.h>
+#include <sys/time.h>
+#endif
```

```

+
+Wheel::Wheel()
+{
+    r = 1;
+    w = 1;
+}
+
+Wheel::Wheel(double x, double y, double z, double radius, double width)
+{
+    r = radius;
+    w = width;
+}
+
+void Wheel::draw()
+{
+    double angle = (rollwheel / r);
+
+    x = getX();
+    y = getY();
+    z = getZ();
+
+    glPushMatrix();
+    glTranslated(x, y, z);
+
+    if (angle != 0)
+    {
+        //translate and then rotate the wheel if the vehicle has moved
+        glTranslated(-(r*sin(angle)), r - r * cos(angle), 0.0);
+        glRotated(-angle * 180 / 3.1415926535, 0.0, 0.0, 1);
+    }
+    glEnd();
+
+    if (getIsSteering() == TRUE)
+    {
+        glRotated(steerwheel, 0, 1, 0);
+    }
+    else
+    {
+        glRotated(rotation, 0.0, 1.0, 0.0);
+    }
+    glColor3d(1,0,0);
+    //fix lighting issues

```

```

+   glDisable(GL_LIGHTING);
+
+   //Following the curved surface drawing function from cylinder to draw the tyre
+   glBegin(GL_TRIANGLE_STRIP);
+
+   for (double angle = 0; angle <= 6.283185307; angle = angle + 0.01)
+   {
+       glVertex3d(r * sin(angle), r+r * cos(angle), (w / 2));
+       w = -w;
+   }
+
+   glEnd();
+   glColor3d(0, 1, 0);
+
+   //Drawing the spoke for the wheel
+   glBegin(GL_QUADS);
+   //bottom
+   glVertex3d(0.1*r, 0, 0.1*r);
+   glVertex3d(0.1*r, 0, -0.1*r);
+   glVertex3d(-0.1*r, 0, -0.1*r);
+   glVertex3d(-0.1*r, 0, 0.1*r);
+
+   //left side
+   glVertex3d(-0.1*r, 0, 0.1*r);
+   glVertex3d(0.1*r, 0, 0.1*r);
+   glVertex3d(0.1*r, 2 * r, 0.1*r);
+   glVertex3d(-0.1*r, 2 * r, 0.1*r);
+
+   //back
+   glVertex3d(-0.1*r, 2 * r, 0.1*r);
+   glVertex3d(-0.1*r, 0, 0.1*r);
+   glVertex3d(-0.1*r, 0, -0.1*r);
+   glVertex3d(-0.1*r, 2 * r, -0.1*r);
+
+   //right
+   glVertex3d(-0.1*r, 2 * r, -0.1*r);
+   glVertex3d(-0.1*r, 0, -0.1*r);
+   glVertex3d(0.1*r, 0, -0.1*r);
+   glVertex3d(0.1*r, 2 * r, -0.1*r);
+
+   //front
+   glVertex3d(0.1*r, 2 * r, -0.1*r);
+   glVertex3d(0.1*r, 0, -0.1*r);

```

```

+     glVertex3d(0.1*r, 0, 0.1*r);
+     glVertex3d(0.1*r, 0 + 2 * r, 0.1*r);
+
+     //top
+     glVertex3d(0.1*r, 2 * r, 0.1*r);
+     glVertex3d(0.1*r, 2 * r,- 0.1*r);
+     glVertex3d(- 0.1*r, 2 * r,- 0.1*r);
+     glVertex3d(- 0.1*r, 2 * r, 0.1*r);
+
+     glEnd();
+
+
+
+     glPopMatrix();
+
+}

```

\ No newline at end of file

--

2.10.5

From bfd1f81f632194952ce07cf50e343c061b06ddbe Mon Sep 17 00:00:00 2001

From: Lachlan <wachlanchow@gmail.com>

Date: Thu, 13 Sep 2018 12:16:47 +1000

Subject: [PATCH] personal\_vehicle\_draw\_functions\_and\_get\_server\_shape\_function

---

Vehicle1.cpp | 468

+++++

Vehicle1.h | 69 +++++++

2 files changed, 537 insertions(+)

create mode 100644 Vehicle1.cpp

create mode 100644 Vehicle1.h

diff --git a/Vehicle1.cpp b/Vehicle1.cpp

new file mode 100644

index 0000000..3ed3dd6

--- /dev/null

+++ b/Vehicle1.cpp

@@ -0,0 +1,468 @@

+#include "Vehicle1.h"

+#include "Wheel.h"

+#include "RectangularPrism.h"

```

#include "TrapezoidalPrism.h"
#include "TriangularPrism.h"
#include "Cylinder.h"
#include "Messages.hpp"
#include <time.h>
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <sstream>
#include <map>
#include <math.h>
+
#ifdef APPLE
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#include <GLUT/glut.h>
#include <unistd.h>
#include <sys/time.h>
#elif defined(WIN32)
#include <Windows.h>
#include <tchar.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#else
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <unistd.h>
#include <sys/time.h>
#endif
+
+Vehicle1::Vehicle1()
+{
+    //body
+    Shape *body1 = new RectangularPrism(0, 0.5, 0, 4, 1, 2);
+    RectangularPrism *rect = dynamic_cast<RectangularPrism*>(body1);
+    body1->setPosition(0, 0.5, 0);
+    body1->setColor(1, 1, 1);
+    body1->setRotation(rotation);
+    addShape(body1);
+

```

```

+ //roof
+ Shape *body2 = new TrapezoidalPrism(0, 1.5, 0, 3, 2, 0.75, 0.5, 2);
+ TrapezoidalPrism *trap = dynamic_cast<TrapezoidalPrism*>(body2);
+ body2->setPosition(0, 1.5, 0);
+ body2->setColor(0, 1, 1);
+ body2->setRotation(rotation);
+ addShape(body2);
+
+ //wheels
+ Shape *body3 = new Wheel(1.25, 0, -1.1, 0.45, 0.25);
+ //Shape *body3 = new Wheel(0, 0, 0, 0.45, 0.25);
+
+ Wheel *cyl1 = dynamic_cast<Wheel*>(body3);
+ body3->setPosition(1.25, 0, -1);
+ body3->setColor(1, 0, 0);
+ body3->setRotation(rotation);
+ //body3->setIsSteering(FALSE);
+
+ if (getSpeed() != 0)
+ {
+     body3->setIsRolling(TRUE);
+ }
+ else
+ {
+     body3->setIsRolling(FALSE);
+ }
+
+ addShape(body3);
+
+ Shape *body4 = new Wheel(1.25, 0, 1.1, 0.45, 0.25);
+ Wheel *cyl2 = dynamic_cast<Wheel*>(body4);
+ body4->setPosition(1.25, 0, 1);
+ body4->setColor(1, 0, 0);
+
+ //body4->setIsSteering(FALSE);
+
+ if (getSpeed() != 0)
+ {
+     body4->setIsRolling(TRUE);
+ }
+ else

```

```

+   {
+       body4->setIsRolling(FALSE);
+   }
+
+   body4->setRotation(rotation);
+   addShape(body4);
+
+
+   Shape *body5 = new Wheel(-1.25, 0, -1.1, 0.55, 0.25);
+   Wheel *cyl3 = dynamic_cast<Wheel*>(body5);
+   body5->setPosition(-1.25, 0, -1);
+   body5->setColor(1, 0, 0);
+
+   if (getSteering() != 0) {
+       body5->setIsSteering(true);
+   }
+   else {
+       body5->setIsSteering(false);
+   }
+
+   if (getSpeed() != 0)
+   {
+       body5->setIsRolling(true);
+   }
+   else
+   {
+       body5->setIsRolling(false);
+   }
+
+   body5->setRotation(rotation);
+   addShape(body5);
+
+   Shape *body6 = new Wheel(-1.25, 0, 1.1, 0.55, 0.25);
+   Wheel *cyl4 = dynamic_cast<Wheel*>(body6);
+   body6->setPosition(-1.25, 0, 1.0);
+   body6->setColor(1, 0, 0);
+
+   if (getSteering() != 0) {
+       body6->setIsSteering(TRUE);
+   }
+   else {
+       body6->setIsSteering(FALSE);
+   }

```



```

+
+   if (getSpeed() != 0)
+   {
+       body6->setIsRolling(true);
+   }
+   else
+   {
+       body6->setIsRolling(false);
+   }
+
+   body6->setRotation(rotation);
+   addShape(body6);
+
+   //spoiler
+   Shape *body7 = new RectangularPrism(-1.875, 1.5, 0, 0.25, 0.25, 3);
+   RectangularPrism *rect2 = dynamic_cast<RectangularPrism*>(body7);
+   body7->setPosition(-1.875, 1.5, 0);
+   body7->setColor(1, 1, 0);
+   body7->setRotation(rotation);
+   addShape(body7);
+
+   //front bumper
+   Shape *body8 = new TriangularPrism(2.25, 0.5, 0, 0.5, 0.35, 2, 90);
+   TriangularPrism *bump = dynamic_cast<TriangularPrism*>(body8);
+   body8->setPosition(2.25, 0.5, 0);
+   body8->setColor(1, 0, 0);
+   body8->setRotation(rotation);
+   addShape(body8);
+
+}
+
+Vehicle1::Vehicle1(VehicleModel vm)
+{
+   for (std::vector<ShapeInit>::iterator it = vm.shapes.begin(); it != vm.shapes.end(); ++it) {
+
+       switch (it->type) {
+       case RECTANGULAR_PRISM:
+       {
+           xrect = it->xyz[0];
+           yrect = it->xyz[1];
+           zrect = it->xyz[2];
+
+           rrect = it->rgb[0];

```

```

+         grect = it->rgb[1];
+         brect = it->rgb[2];
+
+         rorect = it->rotation;
+         xlenrect = it->params.rect.xlen;
+         ylenrect = it->params.rect.ylen;
+         zlenrect = it->params.rect.zlen;
+
+         Shape *shape = new RectangularPrism(xrect, yrect, zrect, xlenrect,
ylenrect, zlenrect);
+         RectangularPrism *rect = dynamic_cast<RectangularPrism*>(shape);
+         shape->setPosition(xrect, yrect, zrect);
+         shape->setRotation(rorect);
+         shape->setColor(rrect, grect, brect);
+         addShape(shape);
+
+         break;
+     }
+
+     case TRIANGULAR_PRISM:
+     {
+         xtri = -(it->xyz[0]);
+         ytri = it->xyz[1];
+         ztri = it->xyz[2];
+
+         rtri = it->rgb[0];
+         gtri = it->rgb[1];
+         btri = it->rgb[2];
+
+         rottri = it->rotation;
+         alentri = it->params.tri.alen;
+         blentri = it->params.tri.blen;
+         depthtri = it->params.tri.depth;
+         angletri = it->params.tri.angle;
+
+         Shape *shape = new TriangularPrism(xtri, ytri, ztri, alentri, blentri,
depthtri, angletri);
+         TriangularPrism *rect = dynamic_cast<TriangularPrism*>(shape);
+         shape->setPosition(xtri, ytri, ztri);
+         shape->setRotation(rottri);
+         shape->setColor(rtri, gtri, btri);
+         addShape(shape);
+

```

```

+         break;
+     }
+
+     case TRAPEZOIDAL_PRISM:
+     {
+         xtrap = it->xyz[0];
+         ytrap = it->xyz[1];
+         ztrap = it->xyz[2];
+
+         rtrap = it->rgb[0];
+         gtrap = it->rgb[1];
+         btrap = it->rgb[2];
+
+         rottrap = it->rotation;
+         alentrap = it->params.trap.alen;
+         blentrap = it->params.trap.blen;
+         heighttrap = it->params.trap.height;
+         aofftrap = it->params.trap.aoff;
+         depthtrap = it->params.trap.depth;
+
+         Shape *shape = new TrapezoidalPrism(xtrap, ytrap, ztrap, alentrap,
+ blentrap, heighttrap, aofftrap, depthtrap);
+         TrapezoidalPrism *rect = dynamic_cast<TrapezoidalPrism*>(shape);
+         shape->setPosition(xtrap, ytrap, ztrap);
+         shape->setRotation(rottrap);
+         shape->setColor(rtrap, gtrap, btrap);
+         addShape(shape);
+
+         break;
+     }
+
+     case CYLINDER:
+     {
+         xcyl = it->xyz[0];
+         ycyl = it->xyz[1];
+         zcyl = it->xyz[2];
+
+         rcyl = it->rgb[0];
+         gcyl = it->rgb[1];
+         bcyl = it->rgb[2];
+
+         rotcyl = it->rotation;
+         radiuscyl = it->params.cyl.radius;

```

```

+         depthcyl = it->params.cyl.depth;
+
+         cylRolling = it->params.cyl.isRolling;
+         cylSteering = it->params.cyl.isSteering;
+
+         Shape *shape = new Cylinder(xcyl, ycyl, zcyl, radiuscyl, depthcyl);
+         Cylinder *rect = dynamic_cast<Cylinder*>(shape);
+         shape->setPosition(xcyl, ycyl, zcyl);
+         shape->setRotation(rotcyl);
+         shape->setIsRolling(cylRolling);
+         shape->setIsSteering(cylSteering);
+         shape->setColor(rcyl, gcyl, bcyl);
+         addShape(shape);
+
+         break;
+     }
+ }
+
+}
+
+void Vehicle1::draw()
+{
+
+    for (std::vector<Shape*>::iterator it = shapes.begin(); it != shapes.end(); ++it) {
+        Wheel *cyl = dynamic_cast<Wheel*>(*it);
+        Cylinder *cylind = dynamic_cast<Cylinder*>(*it);
+        glPushMatrix();
+        //glTranslated(x, y, z);
+        positionInGL();
+
+        if (cylind != NULL && cylind->getIsSteering() == true) {
+            cylind->setRotation(-getSteering());
+        }
+
+        if (cyl != NULL && cyl->getIsSteering() == TRUE)
+        {
+            cyl->setSteerWheel(-getSteering());
+        }
+    }
+}

```



```

+   TriangularPrism G(30, -10, -15, 5, 5, 30, 90);
+   G.setColor(255, 0, 0);
+   G.draw();
+   glPopMatrix();
+
+   glPushMatrix();
+   positionInGL();
+   Wheel H(0, 0, 0, 10, 10);
+   glTranslated(-20, -10, 15);
+   if (getSpeed() > 0) {
+   glRotated(-iTimeElapsed * 0.2, 0.0, 0.0, 1.0);
+   }
+   if (getSpeed() < 0) {
+   glRotated(iTimeElapsed * 0.2, 0.0, 0.0, 1.0);
+   }
+   H.draw();
+   glPopMatrix();
+   //glFlush();
+   //glutPostRedisplay();
+
+
+   glPushMatrix();
+   positionInGL();
+   Wheel I(0, 0, 0, 10, 10);
+   glTranslated(-20, -10, -15);
+   if (getSpeed() > 0) {
+   glRotated(-rot, 0.0, 0.0, 1.0);
+   }
+   if (getSpeed() < 0) {
+   glRotated(rot, 0.0, 0.0, 1.0);
+   }
+   I.draw();
+   glPopMatrix();
+   //glFlush();
+   //glutPostRedisplay();
+
+
+   glPushMatrix();
+   positionInGL();
+   Wheel J(0, 0, 0, 10, 10);
+   J.setRotation(getSteering());
+   glTranslated(17.5, -10, 15);
+   J.draw();

```

```

+   glPopMatrix();
+
+   glPushMatrix();
+   positionInGL();
+   Wheel K(0, 0, 0, 10, 10);
+   K.setRotation(getSteering());
+   glTranslated(17.5, -10, -15);
+   K.draw();
+   glPopMatrix();
+   */
+
+   /*
+   Cylinder D(-20, -10, 15, 10, 10);
+   D.setColor(255, 0, 0);
+   D.draw();
+
+   Cylinder E(-20, -10, -15, 10, 10);
+   E.setColor(255, 0, 0);
+   E.draw();
+
+   Cylinder B(17.5, -10, 15, 10, 10);
+   B.setColor(255, 0, 0);
+   B.setRotation(-getSteering());
+   B.draw();
+
+   Cylinder C(17.5, -10, -15, 10, 10);
+   C.setColor(255, 0, 0);
+   glRotated(rotation, 0, 0, 1);
+   C.setRotation(getSpeed());
+   C.draw();
+
+   glColor3d(0, 255, 255);
+   glBegin(GL_LINES);
+   glVertex3d( 17.5, -25, 20);
+   glVertex3d(17.5, -5, 20);
+   glEnd();
+   */
+
+}
+
+void Vehicle1::update(double dt)

```

```

+{
+    speed = clamp(MAX_BACKWARD_SPEED_MPS, speed,
MAX_FORWARD_SPEED_MPS);
+    steering = clamp(MAX_LEFT_STEERING_DEGS, steering,
MAX_RIGHT_STEERING_DEGS);
+
+    // update position by integrating the speed
+    x += speed * dt * cos(rotation * 3.1415926535 / 180.0);
+    z += speed * dt * sin(rotation * 3.1415926535 / 180.0);
+
+    // update heading
+    rotation += dt * steering * speed;
+
+    while (rotation > 360) rotation -= 360;
+    while (rotation < 0) rotation += 360;
+
+
+    if (fabs(speed) < .1)
+        speed = 0;
+    if (fabs(steering) < .1)
+        steering = 0;
+
+
+
+    distance = distance + speed * dt;
+    //setting it ok
+}
+
+
+
+void Vehicle1::setDistance(double d)
+{
+    distance = d;
+}
+
+double Vehicle1::getDistance()
+{
+    return distance;
+    //std::cout << distance;
+}
diff --git a/Vehicle1.h b/Vehicle1.h
new file mode 100644
index 0000000..f435b8b

```



```

--- /dev/null
+++ b/Vehicle1.h
@@ -0,0 +1,69 @@
+#pragma once
+#include "Vehicle.hpp"
+#include "Shape.hpp"
+#include "Messages.hpp"
+
+class Vehicle1 : public Vehicle
+{
+protected:
+    double xrect;
+    double yrect;
+    double zrect;
+    double xlenrect;
+    double ylenrect;
+    double zlenrect;
+    double rrect;
+    double grect;
+    double brect;
+    double rotrrect;
+
+    double xtri;
+    double ytri;
+    double ztri;
+    double alentri;
+    double blentri;
+    double depthtri;
+    double angletri;
+    double rtri;
+    double gtri;
+    double btri;
+    double rotrtri;
+
+    double xtrap;
+    double ytrap;
+    double ztrap;
+    double alentrap;
+    double blentrap;
+    double heighttrap;
+    double aofftrap;
+    double depthtrap;
+    double rtrap;

```

```

+    double gtrap;
+    double btrap;
+    double rottrap;
+
+    double xcyl;
+    double ycyl;
+    double zcyl;
+    double radiuscyl;
+    double depthcyl;
+    double rcyl;
+    double gcyl;
+    double bcyl;
+    double rotcyl;
+    bool cylRolling;
+    bool cylSteering;
+
+    double distance;
+
+public:
+    Vehicle1();
+    Vehicle1(VehicleModel vm);
+    void draw();
+
+    void update(double dt);
+
+    void setDistance(double d);
+    double getDistance();
+
+};

```

\ No newline at end of file

--

2.10.5



