Assignment 2 : Car Soccer

Nikki Kyllonen | kyllo089 | CSCI4611 | 3/1/2017

Overview

Runs a simplified game of Car Soccer, displaying a single player and a single ball, both of which are restricted to the boundaries of the pitch.

Controls

- `UP`, `DOWN`, `LEFT`, and `RIGHT` arrow keys : control direction the car moves
- `w` : cause the car to jump up 5m
- `s` : cause the car to jump down 5m
- `SPACE` : resets car and ball to initial positions

Implementation

- `void Car::draw()` : calls functions to display body and shadow
- `void Car::reset()` : sets car's member variables to initial values
- `void Car::drawBody(vec3 p)` : draws body of the car with respect to the vector p
  - using `GL_QUAD_STRIP`, first draws a strip of rectangles wrapping around clockwise
  - using `GL_QUADS`, draws the front and back panels of the body
- `void Car::drawShadow(vec3 p)` : draws shadow beneath the car with respect to the vector p
  - uses `GL_BLEND` to give the shadow a semi-transparent look
  - using `GL_POLYGON`, draws a rectangle beneath the car to represent its shadow cast onto the pitch
- `void Ball::draw()` : displays ball according to its position
  - translates the ball to its current position and calls `unitSphere()` accordingly
- `void Ball::drawShadow()` : draws shadow beneath the ball
  - uses `GL_BLEND` to give the shadow a semi-transparent look
  - using `GL_POLYGON`, generates and connects a multitude of vertices, spacing each by 0.25 radians to give the appearance of a smooth edge
- `void Ball::reset()` : sets ball's member variables to initial values
- `CarSoccer::CarSoccer()` : constructs new window and initializes both `car` and `ball`
- `CarSoccer::~CarSoccer()` : destructs the current CarSoccer object
- `void CarSoccer::run()` : looping function that runs the game
  - repeatedly calls `simulate()` and `drawGraphics()`
- `void CarSoccer::simulate(float timeStep)` : main function that updates `car` and `ball` values each frame
  - checks car-to-field collisions to keep `car` above the pitch
  - checks ball-car collisions, determining the after collision velocities
  - checks ball-to-wall collisions; first checking if `ball` collided with a goal before determining the

    - new reflected velocity
  - checks car-to-wall collisions to keep `car` within boundaries
- `void CarSoccer::drawGraphics()` : sets up graphics
  - sets up lighting and camera
  - draws the lines of the pitch, the walls, the `car`, and the `ball` to window
- `void CarSoccer::onKeyDown(SDL_KeyboardEvent &e)` : checks for keyboard input
  - checks for `SPACE`, `w`, and `s`
- `void CarSoccer::drawRect(float h, float w, float topLeftX, float topLeftY, float topLeftZ)`
  - uses `GL_LINE_STRIP` to draw white outline of a rectangle of height `h` and width `w`, with top left coordinate at (`topLeftX`, `topLeftZ`)
- `void CarSoccer::drawCircle(float r, float centerX, float centerZ)`
  - uses `GL_LINE_STRIP` to draw white outline of a circle of radius `r`, centered at (`centerX`, `centerZ`)
  - generates and connects a multitude of vertices, spacing each by 0.25 radians to give the appearance of a smooth line
- `vec3 CarSoccer::atWall(vec3 pos, int radius)` : returns normal of wall object is in contact with; returns zero vector if no wall
  - checks if wall is within the distance `radius` from vector `pos`
- `vec3 CarSoccer::reflectVec(vec3 v, vec3 n)` : returns `v` reflected off of a suface with normal `n`
- `vec3 CarSoccer::updatePosAfterWall(vec3 pos, vec3 wNorm, int radius)` : returns new, off-of-wall vec3 `pos`
  - determines which wall collision occured with
  - uses `radius` to calculate new position vector
- `void CarSoccer::drawGoal(float z, vec3 color)` : draws a goal in the xy plane at `z` using given `color`
  - uses `GL_BLEND` to give the goal a semi-transparent look
  - using `GL_LINES`, draws grid that is 10m tall and 20m wide
- `bool CarSoccer::inGoal(float z)` : check to see if ball is in a goal at `z`

Included Files

`draw.hpp` | `engine.hpp` | `main.cpp` | `README.md` | `README.pdf`