[red42](https://github.com/red42)/[**HTML5\_Genetic\_Cars**](https://github.com/red42/HTML5_Genetic_Cars)

[**HTML5\_Genetic\_Cars**](https://github.com/red42/HTML5_Genetic_Cars/tree/02c84ffed612ec93ff4a4ffc9a4b4c691430627b) / **cawro.js**

*// Global Vars*

**var** ghost;

**var** timeStep **=** 1.0 **/** 60.0;

**var** doDraw **=** **true**;

**var** cw\_paused **=** **false**;

**var** box2dfps **=** 60;

**var** screenfps **=** 60;

**var** debugbox **=** document.getElementById("debug");

**var** canvas **=** document.getElementById("mainbox");

**var** ctx **=** canvas.getContext("2d");

//Camera variables

**var** cameraspeed **=** 0.05;

**var** camera\_y **=** 0;

**var** camera\_x **=** 0;

**var** camera\_target **=** **-**1; *// which car should we follow? -1 = leader*

**var** minimapcamera **=** document.getElementById("minimapcamera").style;

//environment restrictioins

**var** graphcanvas **=** document.getElementById("graphcanvas");

**var** graphctx **=** graphcanvas.getContext("2d");

**var** graphheight **=** 250;

**var** graphwidth **=** 400;

**var** minimapcanvas **=** document.getElementById("minimap");

**var** minimapctx **=** minimapcanvas.getContext("2d");

**var** minimapscale **=** 3;

**var** minimapfogdistance **=** 0;

**var** fogdistance **=** document.getElementById("minimapfog").style;

//array declarations

**var** generationSize **=** 20;

**var** cw\_carArray **=** **new** Array();

**var** cw\_carGeneration **=** **new** Array();

**var** cw\_carScores **=** **new** Array();

**var** cw\_topScores **=** **new** Array();

**var** cw\_graphTop **=** **new** Array();

**var** cw\_graphElite **=** **new** Array();

**var** cw\_graphAverage **=** **new** Array();

**var** gen\_champions **=** 1;

**var** gen\_parentality **=** 0.2;

**var** gen\_mutation **=** 0.05;

**var** mutation\_range **=** 1;

**var** gen\_counter **=** 0;

**var** nAttributes **=** 15;

**var** gravity **=** **new** b2Vec2(0.0, **-**9.81);

**var** doSleep **=** **true**;

**var** world;

**var** zoom **=** 70;

**var** mutable\_floor **=** **false**;

**var** maxFloorTiles **=** 200;

**var** cw\_floorTiles **=** **new** Array();

**var** last\_drawn\_tile **=** 0;

**var** groundPieceWidth **=** 1.5;

**var** groundPieceHeight **=** 0.15;

//car dimensions restrictions

**var** chassisMaxAxis **=** 1.1;

**var** chassisMinAxis **=** 0.1;

**var** chassisMinDensity **=** 30;

**var** chassisMaxDensity **=** 300;

//wheel restrictioins

**var** wheelMaxRadius **=** 0.5;

**var** wheelMinRadius **=** 0.2;

**var** wheelMaxDensity **=** 100;

**var** wheelMinDensity **=** 40;

//car movement and health

**var** velocityIndex **=** 0;

**var** deathSpeed **=** 0.1;

**var** max\_car\_health **=** box2dfps **\*** 10;

**var** car\_health **=** max\_car\_health;

**var** motorSpeed **=** 20;

**var** swapPoint1 **=** 0;

**var** swapPoint2 **=** 0;

**var** cw\_ghostReplayInterval **=** **null**;

**var** distanceMeter **=** document.getElementById("distancemeter");

**var** leaderPosition **=** **new** Object();

leaderPosition.x **=** 0;

leaderPosition.y **=** 0;

minimapcamera.width **=** 12**\***minimapscale**+**"px";

minimapcamera.height **=** 6**\***minimapscale**+**"px";

//start here

**function** debug(str, clear) {

**if**(clear) {

debugbox.innerHTML **=** "";

}

debugbox.innerHTML **+=** str**+**"<br />";

}

**function** showDistance(distance, height) {

distanceMeter.innerHTML **=** "distance: "**+**distance**+**" meters<br />";

distanceMeter.innerHTML **+=** "height: "**+**height**+**" meters";

**if**(distance **>** minimapfogdistance) {

fogdistance.width **=** 800 **-** Math.round(distance **+** 15) **\*** minimapscale **+** "px";

minimapfogdistance **=** distance;

}

}

*/\* ========================================================================= \*/*

*/\* === Car ================================================================= \*/*

*/\*sets up velocity health and position of car*

*\*wheels*

*\*chassis*

*\*/*

**var** cw\_Car **=** **function**() {

**this**.\_\_constructor.apply(**this**, arguments);

}

cw\_Car.prototype.chassis **=** **null**;

cw\_Car.prototype.wheels **=** [];

cw\_Car.prototype.\_\_constructor **=** **function**(car\_def) {

**this**.velocityIndex **=** 0;

**this**.health **=** max\_car\_health;

**this**.maxPosition **=** 0;

**this**.maxPositiony **=** 0;

**this**.minPositiony **=** 0;

**this**.frames **=** 0;

**this**.car\_def **=** car\_def

**this**.alive **=** **true**;

**this**.is\_elite **=** car\_def.is\_elite;

**this**.healthBar **=** document.getElementById("health"**+**car\_def.index).style;

**this**.healthBarText **=** document.getElementById("health"**+**car\_def.index).nextSibling.nextSibling;

**this**.healthBarText.innerHTML **=** car\_def.index;

**this**.minimapmarker **=** document.getElementById("bar"**+**car\_def.index);

**if**(**this**.is\_elite) {

**this**.healthBar.backgroundColor **=** "#44c";

**this**.minimapmarker.style.borderLeft **=** "1px solid #44c";

**this**.minimapmarker.innerHTML **=** car\_def.index;

} **else** {

**this**.healthBar.backgroundColor **=** "#c44";

**this**.minimapmarker.style.borderLeft **=** "1px solid #c44";

**this**.minimapmarker.innerHTML **=** car\_def.index;

}

**this**.chassis **=** cw\_createChassis(car\_def.vertex\_list, car\_def.chassis\_density);

**this**.wheels **=** [];

**for** (**var** i **=** 0; i **<** car\_def.wheelCount; i**++**){

**this**.wheels[i] **=** cw\_createWheel(car\_def.wheel\_radius[i], car\_def.wheel\_density[i]);

}

**var** carmass **=** **this**.chassis.GetMass();

**for** (**var** i **=** 0; i **<** car\_def.wheelCount; i**++**){

carmass **+=** **this**.wheels[i].GetMass();

}

**var** torque **=** [];

**for** (**var** i **=** 0; i **<** car\_def.wheelCount; i**++**){

torque[i] **=** carmass **\*** **-**gravity.y **/** car\_def.wheel\_radius[i];

}

**var** joint\_def **=** **new** b2RevoluteJointDef();

**for** (**var** i **=** 0; i **<** car\_def.wheelCount; i**++**){

**var** randvertex **=** **this**.chassis.vertex\_list[car\_def.wheel\_vertex[i]];

joint\_def.localAnchorA.Set(randvertex.x, randvertex.y);

joint\_def.localAnchorB.Set(0, 0);

joint\_def.maxMotorTorque **=** torque[i];

joint\_def.motorSpeed **=** **-**motorSpeed;

joint\_def.enableMotor **=** **true**;

joint\_def.bodyA **=** **this**.chassis;

joint\_def.bodyB **=** **this**.wheels[i];

**var** joint **=** world.CreateJoint(joint\_def);

}

**this**.replay **=** ghost\_create\_replay();

ghost\_add\_replay\_frame(**this**.replay, **this**);

}

cw\_Car.prototype.getPosition **=** **function**() {

**return** **this**.chassis.GetPosition();

}

cw\_Car.prototype.draw **=** **function**() {

drawObject(**this**.chassis);

**for** (**var** i **=** 0; i **<** **this**.wheels.length; i**++**){

drawObject(**this**.wheels[i]);

}

}

cw\_Car.prototype.kill **=** **function**() {

**var** avgspeed **=** (**this**.maxPosition **/** **this**.frames) **\*** box2dfps;

**var** position **=** **this**.maxPosition;

**var** score **=** position **+** avgspeed;

ghost\_compare\_to\_replay(**this**.replay, ghost, score);

cw\_carScores.push({ car\_def**:this**.car\_def, v**:**score, s**:** avgspeed, x**:**position, y**:this**.maxPositiony, y2**:this**.minPositiony });

world.DestroyBody(**this**.chassis);

**for** (**var** i **=** 0; i **<** **this**.wheels.length; i**++**){

world.DestroyBody(**this**.wheels[i]);

}

**this**.alive **=** **false**;

*// refocus camera to leader on death*

**if** (camera\_target **==** **this**.car\_def.index){

cw\_setCameraTarget(**-**1);

}

}

cw\_Car.prototype.checkDeath **=** **function**() {

*// check health*

**var** position **=** **this**.getPosition();

**if**(position.y **>** **this**.maxPositiony) {

**this**.maxPositiony **=** position.y;

}

**if**(position .y **<** **this**.minPositiony) {

**this**.minPositiony **=** position.y;

}

**if**(position.x **>** **this**.maxPosition **+** 0.02) {

**this**.health **=** max\_car\_health;

**this**.maxPosition **=** position.x;

} **else** {

**if**(position.x **>** **this**.maxPosition) {

**this**.maxPosition **=** position.x;

}

**if**(Math.abs(**this**.chassis.GetLinearVelocity().x) **<** 0.001) {

**this**.health **-=** 5;

}

**this**.health**--**;

**if**(**this**.health **<=** 0) {

**this**.healthBarText.innerHTML **=** "&#8708;";

**this**.healthBar.width **=** "0";

**return** **true**;

}

}

}

/\*

\*

\*/

**function** cw\_createChassisPart(body, vertex1, vertex2, density) {

**var** vertex\_list **=** **new** Array();

vertex\_list.push(vertex1);

vertex\_list.push(vertex2);

vertex\_list.push(b2Vec2.Make(0,0));

**var** fix\_def **=** **new** b2FixtureDef();

fix\_def.shape **=** **new** b2PolygonShape();

fix\_def.density **=** density;

fix\_def.friction **=** 10;

fix\_def.restitution **=** 0.2;

fix\_def.filter.groupIndex **=** **-**1;

fix\_def.shape.SetAsArray(vertex\_list,3);

body.CreateFixture(fix\_def);

}

//creats the eight segments of the car

**function** cw\_createChassis(vertex\_list, density) {

**var** body\_def **=** **new** b2BodyDef();

body\_def.type **=** b2Body.b2\_dynamicBody;

body\_def.position.Set(0.0, 4.0);

**var** body **=** world.CreateBody(body\_def);

cw\_createChassisPart(body, vertex\_list[0],vertex\_list[1], density);

cw\_createChassisPart(body, vertex\_list[1],vertex\_list[2], density);

cw\_createChassisPart(body, vertex\_list[2],vertex\_list[3], density);

cw\_createChassisPart(body, vertex\_list[3],vertex\_list[4], density);

cw\_createChassisPart(body, vertex\_list[4],vertex\_list[5], density);

cw\_createChassisPart(body, vertex\_list[5],vertex\_list[6], density);

cw\_createChassisPart(body, vertex\_list[6],vertex\_list[7], density);

cw\_createChassisPart(body, vertex\_list[7],vertex\_list[0], density);

body.vertex\_list **=** vertex\_list;

**return** body;

}

**function** cw\_createWheel(radius, density) {

**var** body\_def **=** **new** b2BodyDef();

body\_def.type **=** b2Body.b2\_dynamicBody;

body\_def.position.Set(0, 0);

**var** body **=** world.CreateBody(body\_def);

**var** fix\_def **=** **new** b2FixtureDef();

fix\_def.shape **=** **new** b2CircleShape(radius);

fix\_def.density **=** density;

fix\_def.friction **=** 1;

fix\_def.restitution **=** 0.2;

fix\_def.filter.groupIndex **=** **-**1;

body.CreateFixture(fix\_def);

**return** body;

}

//Creates random car

**function** cw\_createRandomCar() {

**var** v **=** [];

**var** car\_def **=** **new** Object();

car\_def.wheelCount **=** 2;

car\_def.wheel\_radius **=** [];

car\_def.wheel\_density **=** [];

car\_def.wheel\_vertex **=** [];

**for** (**var** i **=** 0; i **<** car\_def.wheelCount; i**++**){

car\_def.wheel\_radius[i] **=** Math.random()**\***wheelMaxRadius**+**wheelMinRadius;

car\_def.wheel\_density[i] **=** Math.random()**\***wheelMaxDensity**+**wheelMinDensity;

}

car\_def.chassis\_density **=** Math.random()**\***chassisMaxDensity**+**chassisMinDensity

car\_def.vertex\_list **=** **new** Array();

car\_def.vertex\_list.push(**new** b2Vec2(Math.random()**\***chassisMaxAxis **+** chassisMinAxis,0));

car\_def.vertex\_list.push(**new** b2Vec2(Math.random()**\***chassisMaxAxis **+** chassisMinAxis,Math.random()**\***chassisMaxAxis **+** chassisMinAxis));

car\_def.vertex\_list.push(**new** b2Vec2(0,Math.random()**\***chassisMaxAxis **+** chassisMinAxis));

car\_def.vertex\_list.push(**new** b2Vec2(**-**Math.random()**\***chassisMaxAxis **-** chassisMinAxis,Math.random()**\***chassisMaxAxis **+** chassisMinAxis));

car\_def.vertex\_list.push(**new** b2Vec2(**-**Math.random()**\***chassisMaxAxis **-** chassisMinAxis,0));

car\_def.vertex\_list.push(**new** b2Vec2(**-**Math.random()**\***chassisMaxAxis **-** chassisMinAxis,**-**Math.random()**\***chassisMaxAxis **-** chassisMinAxis));

car\_def.vertex\_list.push(**new** b2Vec2(0,**-**Math.random()**\***chassisMaxAxis **-** chassisMinAxis));

car\_def.vertex\_list.push(**new** b2Vec2(Math.random()**\***chassisMaxAxis **+** chassisMinAxis,**-**Math.random()**\***chassisMaxAxis **-** chassisMinAxis));

**var** left **=** [];

**for** (**var** i **=** 0; i **<** 8; i**++**){

left.push(i);

}

**for** (**var** i **=** 0; i **<** car\_def.wheelCount; i**++**){

**var** indexOfNext **=** Math.floor(Math.random()**\***left.length);

car\_def.wheel\_vertex[i] **=** left[indexOfNext];

left.splice(indexOfNext, 1);

}

**return** car\_def;

}

*/\* === END Car ============================================================= \*/*

*/\* ========================================================================= \*/*