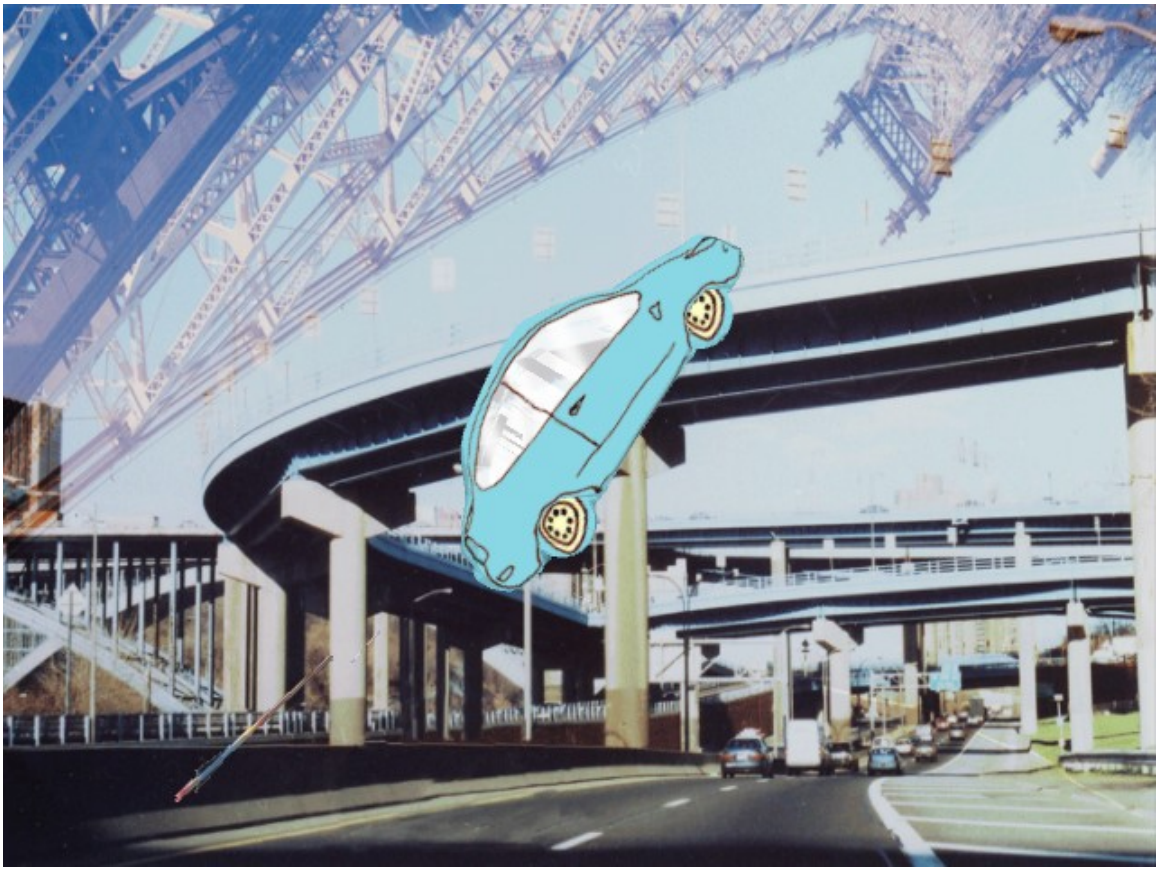
Le jeu *Kill a Car*



Myriam Bizier 2009, Studio XX

Le jeu *Kill a Car*

Intention

Tuons le char! Le jeu *Kill a Car* affirme que le symbole de la Voiture est désuet et qu'il devrait mourir. Tuons l'automobile, ce vecteur d'apparence et d'égo, cet outil de l'individualisme et cette occasion d'excès. Tuons la voiture, cette banalité, cette norme qui, à l'accumulation, a modelé notre idée du vivre au quotidien et de nos espaces communs. Tuons le char, ce contenant de taule à options multiples qui sort de l'usine en ayant laissé derrière lui vingt-cinq tonnes de déchets et détourné une rivière d'eau. Et tuons le pétrole! Lien de cause à effet crucial dans l'objet comme dans le symbol Voiture, le pétrole change de prix à tous les jours. Ici, avec le jeu*Kill a Car*, plus le pétrole au baril est cher, plus la voiture s'enlise.

Inspirations

Kill a Car est inspirée par des œuvres mettant en scène des véhicules, lesquelles m'ont toujours touchée à travers les années. J'ai remarqué que ces œuvrestendent à apparaître avec plus d'insistance depuis les années 2000, dont entre autres la trilogie vidéo *Chrysalides* de Patrick Bernatchez, les installations *Inoportunes* de Cai Guo-Qian, *Holidays* de Pierre Ardouvin et *Char Brun* de Stéphanie Pelletier, mais surtout par l' œuvre*Jouet d'adulte* du collectif BGL.

Chrysalides
Patrick Bernatchez
2008

Holidays
Pierre Ardouvin
1999

Char Brun
Stéphanie Pelletier
2006

Inoportunes
Cai Guo-Qian
2008

Jouet d'adulte
BGL
2003

D'une part Bernatchez et Ardouvin présentent la voiture comme un centre de pivot, mis en scène à la fois comme célébrité et échec: l'une est encerclée par une caméra insistante et l'autre tourne sur elle-même dans une atmosphère de fête. D'autre part, les œuvres de Guo-Cian et de BGL présentent des véhicules piqués d'explosifs et de flèches, comme des extensions imaginaires de leur propres volumes de taule. Plus littéralement, *Jouet d'Adulte* présente le quatre-roues comme un objet de chasse tribale, renversant l'image du chasseur sur son véhicule.

Questionner l'omniprésence de la voiture

consoglobe.com

Le jeu *Kill a Car* est donc une synthèse de ces deux états d'esprit et prend les spécificités du 3D et de l'interactivité pour faire passer une rage contre le culte de la voiture. En terme de transport, d'urbanisme et de mode de vie, il est grand temps de changements pour les citoyens et les états: nous devons penser en fonction des transports actifs et collectifs pour augmenter nos qualité de vie et celles des générations devant nous

Industrie pétrolière canadienne; des liens de cause à effet et des hyperliens

syncrude.ca

Au Canada, une certaine industrie de l'énergie est loin d'être un empêcheur de tourner en rond dans le domaine du transport. Les sables bitumineux exploités en Alberta consituent une réserve importante de pétrole, et le gouvernement en place est plus que favorable à cette activité et aux marchés qui lui sont rattachés.

Des compagnies comme Syncrude et Total, en lien avec le Canadian Oil Sands Trust, produisent le pétrole nommé Western Canadian Select. Elles opèrent loin de nos regards, isolée, aux endroits même où des communautés autochtones voient leur rivières, source de nourriture, s'empoisonner par l'exploitation pétrolière et sont contraints de travailler sur ces chantiers pour se maintenir en vie. L'exploitation pétrolière en Alberta se paie d'énormes coût environnementaux et sociaux. Des hyperliens sont donc suggérés sur le site de *Kill a Car* pour mettre en perspective le point de vue des corporations et celui des reporter de VBS.tv dans leur série Toxic Alberta.

Cheminement technique

- ✔ Un wiki pour garder le cap et partager nos acquis.

  Monté avec Alexandre Quessy, consultant en programmation. Merci Alex pour l'initiative! On trouve sur cette page notre liste de tâches priorisées, des versions de travail antérieures et quelques ressources en hyperliens.

- ✔ Le choix d'une diffusion web, donc de l'outil Processing

  Avec l'atelier en ludologie numérique, nous avons appris surtout Blender. Il est venu un moment où on nous a suggéré d'autres moyens, d'autres outils, comme Processing. J'ai donc décidé d'apprendre Processing pour être en mesure de diffuser un projet à contenu engagé de manière assez accessible, à même un site web où l'on peut jouer, ce que Blender ne permet pas. Processing est un éditeur de programmation en Java qui est optimisé pour les artistes et plutôt accessible d'utilisation sans trop de connaissances poussées. Il permet de faire des animations visuelles interactive d'une grande qualités visuelle. Il est parfois combiné à OpenGL

- ✔ Intégrer des objets 3D dans Processing

  Il existe une librairie nommée Objloader qui permet d'importer dans Processing des « mesh » 3D construits dans Blender et exporté en .obj. Pour que les matériaux et mapping de texture se rendent bien, il faut, dans Blender, les appliquer comme on le ferait pour le moteur de jeu, et non pas comme on le ferait pour un rendu en 2D.

- ✔ Programmer les collisions dans un univers en 3 dimensions: Quaternions, simulation de physique: le choix de JBullet

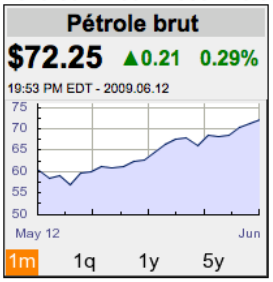
  JBullet est un ensemble de librairies en Java qui permet de simuler les lois de la physique dans un univers en 3d. Ces calculs de précision sont assurés par des librairies comme Bullet. Bullet est la version officielle de cet outil, en C++. Elle intègre des calculs de détection de collision et des contraintes de mouvements, ce dont le jeu a besoin pour faire coller les flèches à la voiture. JBullet a été porté à Processing par Alexandre Quessy.

- ✔ Faire pivoter la voiture en apesanteur, dans un monde à gravité.

  Vélocités angulaires pour faire pivoter la voiture et vélocité linéaire vers le haut pour contrer la gravité.

- ✔ Intégrer la variable du prix du pétrole

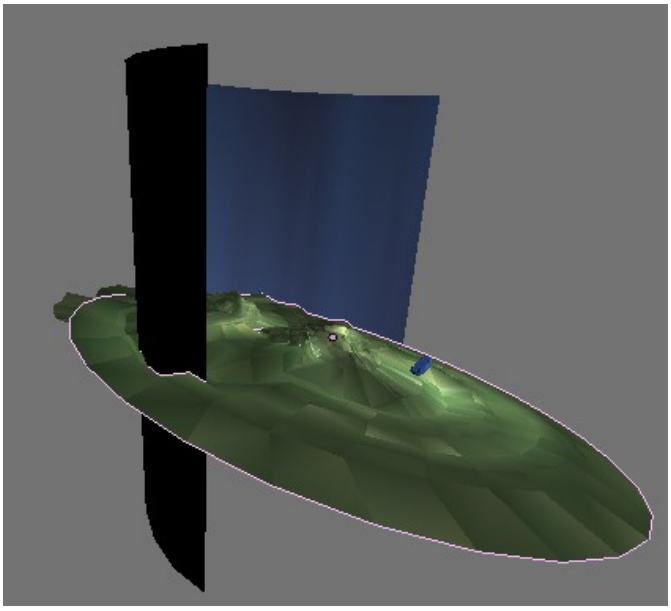
  Appeler la valeur du prix du pétrole brut trouvée sur un site internet et la traiter en PHP pour influencer le jeu grâce à la classe « param » de Processing, qui renvois au tag html param d'une application imbriquée sur une page web.

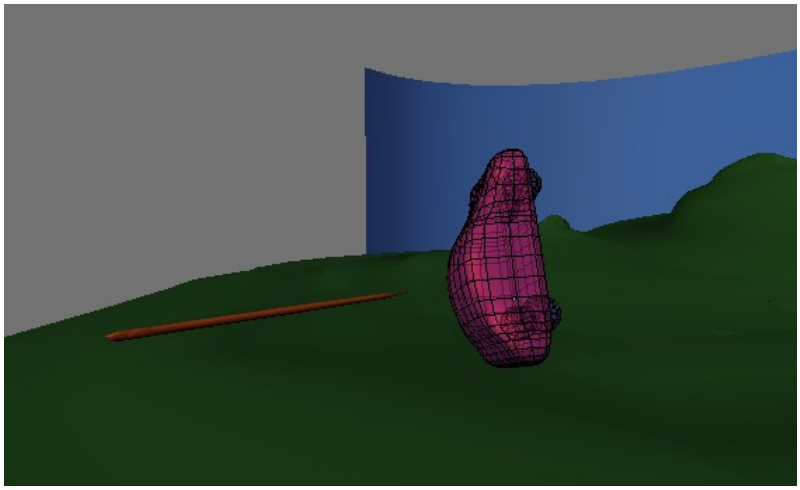


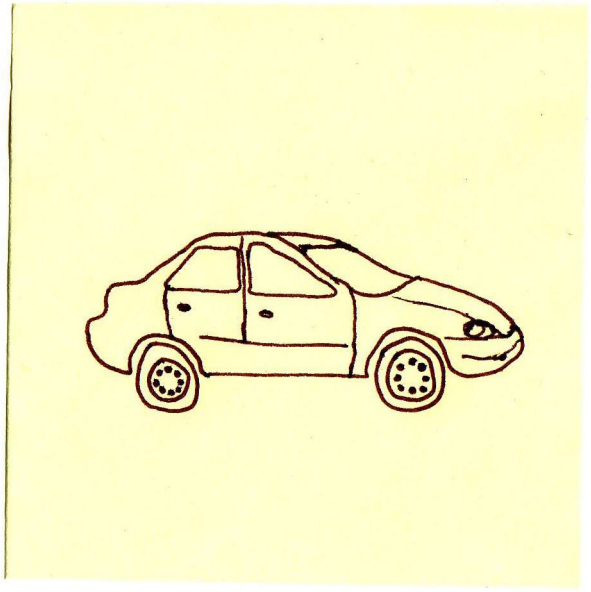
- ✔ Recherches sur les classes de Bullet qui répondent aux besoins spécifiques du jeu *Kill a car*

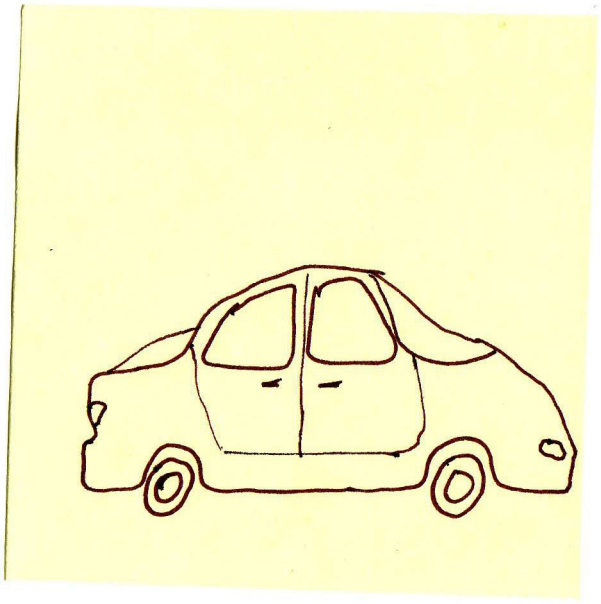
  - Détecter des collisions: une certaine classe ou méthode «Manifolds » encore à l'étape de recherche;
  - Appliquer une contrainte à l'objet flèche pour qu'elle reste attachée à l'objet voiture après la collision: btConeTwistConstraint, à activer lors de la détection de collision. Ceci reste à faire.

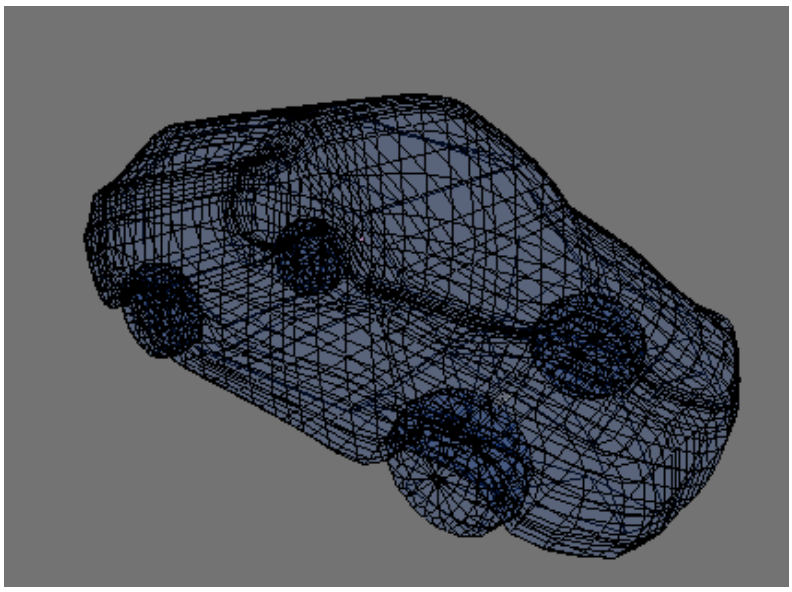
***Kill A Car*, quelques apperçus du cheminement**

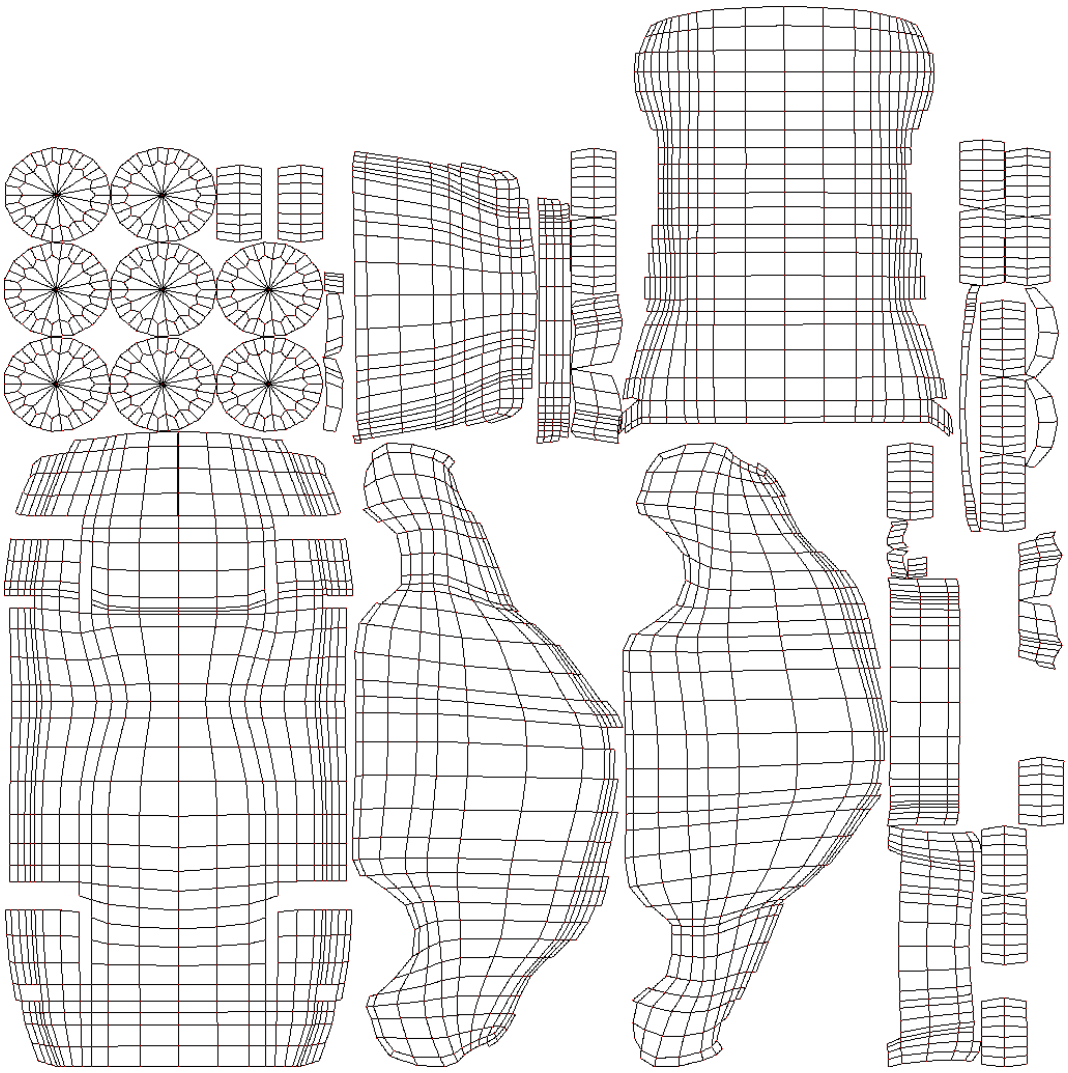


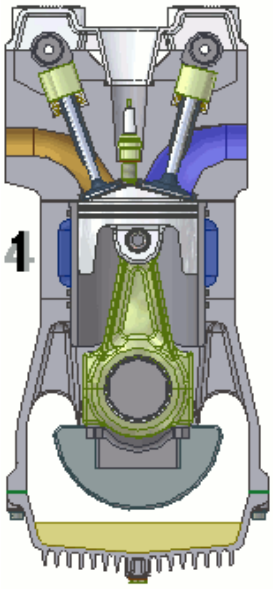


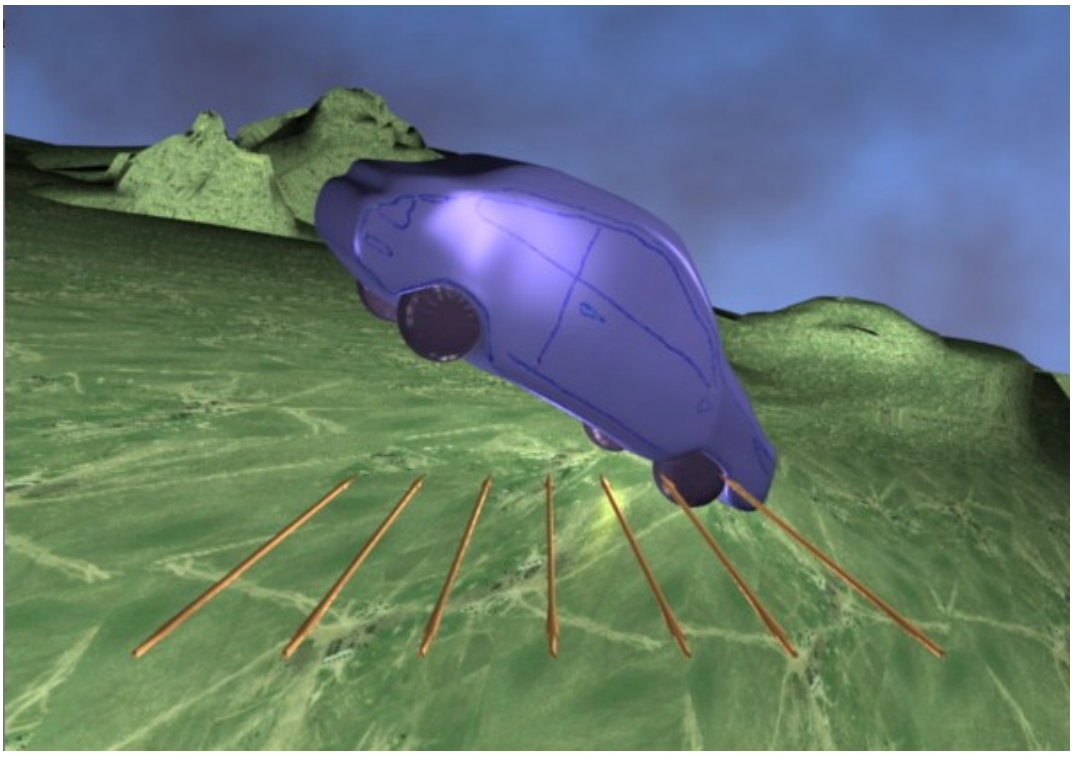


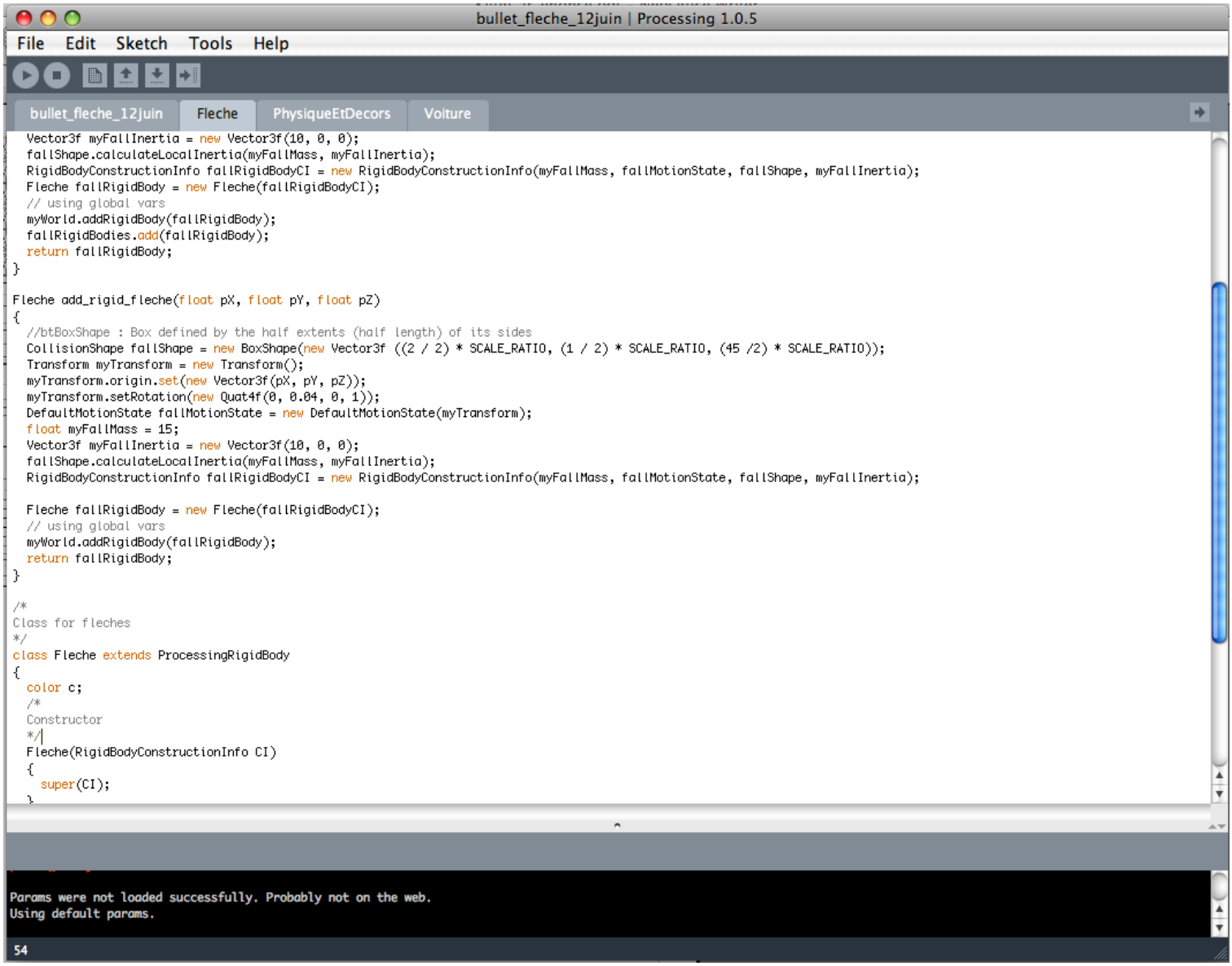


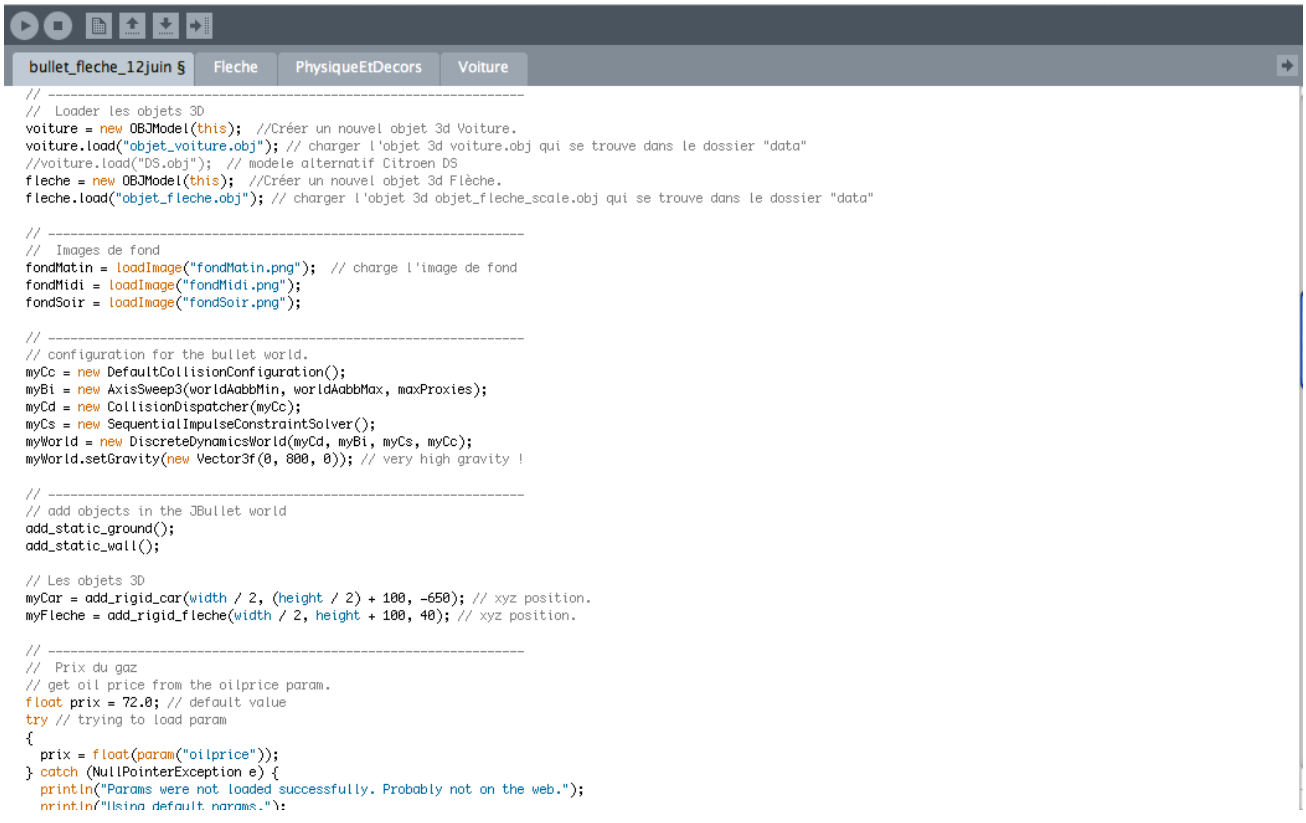





1




bullet_fleche_12juin | Processing 1.0.5
File Edit Sketch Tools Help
bullet_fleche_12juin
Fleche
PhysiqueEtDecors
Voiture
Vector3f myFallInertia = new Vector3f(10, 0, 0);
fallShape.calculateLocalInertia(myFallMass, myFallInertia);
RigidBodyConstructionInfo fallRigidBodyCI = new RigidBodyConstructionInfo(myFallMass, fallMotionState, fallShape, myFallInertia);
Fleche fallRigidBody = new Fleche(fallRigidBodyCI);
// using global vars
myWorld.addRigidBody(fallRigidBody);
fallRigidBodies.add(fallRigidBody);
return fallRigidBody;
}
Fleche add_rigid_fleche(float pX, float pY, float pZ)
{
//btBoxShape : Box defined by the half extents (half length) of its sides
CollisionShape fallShape = new BoxShape(new Vector3f ((2 / 2) * SCALE_RATIO, (1 / 2) * SCALE_RATIO, (45 /2) * SCALE_RATIO));
Transform myTransform = new Transform();
myTransform.origin.set(new Vector3f(pX, pY, pZ));
myTransform.setRotation(new Quat4f(0, 0.04, 0, 1));
DefaultMotionState fallMotionState = new DefaultMotionState(myTransform);
float myFallMass = 15;
Vector3f myFallInertia = new Vector3f(10, 0, 0);
fallShape.calculateLocalInertia(myFallMass, myFallInertia);
RigidBodyConstructionInfo fallRigidBodyCI = new RigidBodyConstructionInfo(myFallMass, fallMotionState, fallShape, myFallInertia);
Fleche fallRigidBody = new Fleche(fallRigidBodyCI);
// using global vars
myWorld.addRigidBody(fallRigidBody);
return fallRigidBody;
}
/*
Class for fleches
*/
class Fleche extends ProcessingRigidBody
{
color c;
/*
Constructor
*/
Fleche(RigidBodyConstructionInfo CI)
{
super(CI);
}
Params were not loaded successfully. Probably not on the web.
Using default params.
54


bullet_fleche_12juin §
Fleche
PhysiqueEtDecors
Voiture
// ----------
// Loader les objets 3D
voiture = new OBJModel(this); //Créer un nouvel objet 3d Voiture.
voiture.load("objet_voiture.obj"); // charger l'objet 3d voiture.obj qui se trouve dans le dossier "data"
//voiture.load("DS.obj"); // modele alternatif Citroen DS
fleche = new OBJModel(this); //Créer un nouvel objet 3d Flèche.
fleche.load("objet_fleche.obj"); // charger l'objet 3d objet_fleche_scale.obj qui se trouve dans le dossier "data"
// ----------
// Images de fond
fondMatin = loadImage("fondMatin.png"); // charge l'image de fond
fondMidi = loadImage("fondMidi.png");
fondSoir = loadImage("fondSoir.png");
// ----------
// configuration for the bullet world.
myCc = new DefaultCollisionConfiguration();
myBi = new AxisSweep3(worldAabbMin, worldAabbMax, maxProxies);
myCd = new CollisionDispatcher(myCc);
myCs = new SequentialImpulseConstraintSolver();
myWorld = new DiscreteDynamicsWorld(myCd, myBi, myCs, myCc);
myWorld.setGravity(new Vector3f(0, 800, 0)); // very high gravity !
// ----------
// add objects in the JBullet world
add_static_ground();
add_static_wall();
// Les objets 3D
myCar = add_rigid_car(width / 2, (height / 2) + 100, -650); // xyz position.
myFleche = add_rigid_fleche(width / 2, height + 100, 40); // xyz position.
// ----------
// Prix du gaz
// get oil price from the oilprice param.
float prix = 72.0; // default value
try // trying to load param
{
prix = float(param("oilprice"));
} catch (NullPointerException e) {
println("Params were not loaded successfully. Probably not on the web.");
println("Using default params.");