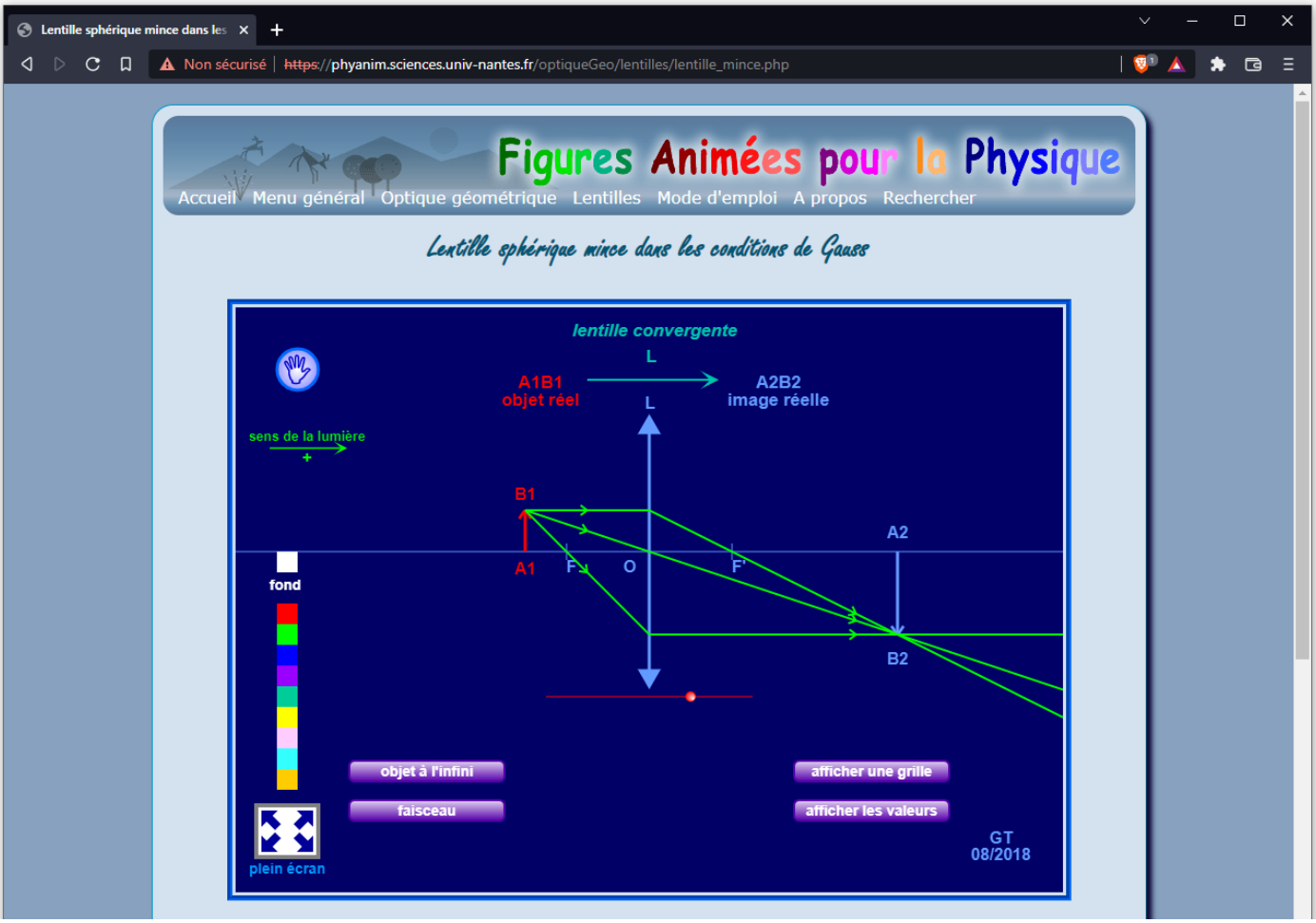
Rapport de stage

Sujet : Amélioration d'une simulation web d'une lentille



pour la période du 11 Avril au 25 Mai 2022

Auteurs : Sylvain Augé, Abdelilah Tahiri

Enseignants encadrants : Benoît Crespin, Julien Brevier

Université de Limoges

Faculté des Sciences & Techniques

Dédicace

En témoignage de l’amour, de l’affection, du respect et de l'attachement, on dédie chaleureusement ce modeste travail et cet événement marquant de nos vies à :

Nos très chers parents pour leurs parrainages, affection et amour, leur confiance, patience et surtout pour leurs engagements fournis jour et nuit pour notre éducation et notre bien-être. Ce travail n’est qu’un miroir de leurs efforts et sacrifices infinis.

Nos professeurs, pour leur soutient, leurs encouragements, leur sérieux, leurs attentions et leur sens du devoir qui sont l’origine de notre réussite.

Remerciements

Avant d’aborder la rédaction de notre rapport de stage, nous tenons profondément à remercier tous ceux qui ont contribué à la réussite de ce stage, et qui n’ont hésité en aucun moment à nous fournir les informations nécessaires concernant le déroulement de notre stage.

Nous tenons profondément à remercier nos deux professeurs responsables, M. Crespin et M. Brevier qui nous ont fourni les outils nécessaires et qu’ils n’ont pas cessé de déployer pour mener à bien ce travail. Nous voulons remercier spécialement M. Olivier Terraz, qui fut le premier à nous soutenir dans la démarche de stage.

Nous saisissons cette occasion pour adresser nos profonds remerciements aux responsables et au personnel à la direction de la faculté, la faculté des sciences et techniques de Limoges qui nous a donné l’opportunité de réaliser ce stage. Nos vifs remerciements s’adressent également à toute personne qui a contribué d’une façon ou d’une autre, de près ou de loin, à l’accomplissement de ce projet.

Sommaire

I – Présentation du sujet

Le projet qui nous a été confié au cours des deux derniers mois est la modification d'un site internet proposant une simulation du comportement de la lumière au travers d'une lentille.
Ce site appartient à l'université de Nantes et regroupe de nombreuses pages similaires permettant d'observer des phénomènes physiques liés à la mécanique ou bien à l'électronique.
Il est aussi utilisé au sein de l'université de Limoges dans le cadre de cours sur l'optique.
Monsieur Brevier, l'un de nos encadrants, est enseignant en optique et se sert de cette lentille pour illustrer à ces élèves les phénomènes de diffractions de la lumière.

Elle intègre différentes fonctionnalités qui permettent de visualiser comment se comporte la trajectoire d'ondes lumineuses lorsqu'elles traversent une lentille sphérique en verre.
Il est possible de déplacer plusieurs éléments sur la fenêtre afin d'avoir des résultats selon des cas différents. Par exemple, il est possible de travailler sur une lentille aux bords concaves qui sera convergente ; ou aux bords convexes, qui sera divergente.

Cependant, la liberté qui est accordée dans les déplacements peut se révéler trompeuse car elle offre une multitude de scénarios en permanence. Étant utilisée dans des exercices où il n'est nécessaire d'avoir que quelques cas de figures à chaque fois, la possibilité de restreindre les mouvements permettrait de se concentrer uniquement sur ceux qui sont visés.
C'est de cette idée de Monsieur Brevier qu'est venu notre sujet de stage.

Nous avons donc participé ces derniers mois à l'amélioration de cette simulation en travaillant sur la concrétisation de ce concept ainsi que d'autres ajouts qui ont été fait tout le long du projet. Après une rapide analyse du code existant, nous traiterons dans ce rapport du formulaire qui a été créé pour fixer des limites ; puis de toutes les fonctionnalités qui sont apparues en plus.

Déroulement du stage

Depuis les premières réunions avec nos deux encadrants, nous nous sommes mis d'accord pour que dans un premier temps chacun développe, de son côté, un script qui sera capable d'avoir les résultats attendus par notre client : Monsieur Brevier. Chose qui a créé une sorte de dynamicité. Chacun a œuvré en vue d'atteindre des objectifs communs, et vu que l'on a travaillé pour le même but : on était très soudé. Le fait de travailler en binôme nous a permis de progresser et nous a beaucoup apporté. On s'est entraidé pour que l'on progresse dans le projet de deux manières complètement différentes.

Nous nous sommes répartit les tâches pour ne pas travailler sur les mêmes fonctionnalités durant la première moitié du stage. Nous avons malgré tout chacun proposé une version du formulaire pour pouvoir être mieux évalué indépendamment et garder le meilleur de ce qui était fait. On a ensuite fusionné nos deux projets pour n'avoir qu'un seul formulaire et continué à avancer sur différentes missions en parallèle.

Technologies utilisées

L’informatique est un secteur d’activité bien composite, que ce soit au niveau du développement ou de la maintenance. Pour nous mettre sur la bonne route et mettre notre travail en adéquation avec le script déjà développé, on a projeté nos aptitudes en employant des technologies déjà utilisées qui se présentent comme suit :

HTML : L’acronyme d’HyperText Markup Langage, c’est la partie du code qui se charge de structurer le contenu d’une page web. Il se compose d'un enchaînement d’éléments appelés balise qui est ordonnancé de façon à permettre un contrôle d’affichage ou de saisie du contenu de la page web.

CSS : L’acronyme de Cascading Style Sheets, c’est un langage informatique utilisé sur internet pour appliquer une apparence soignée et une mise en forme des fichiers et des pages HTML simultanément.

JavaScript : désigne un langage de développement informatique qui se présente en complément de HTML et CSS. Il permet principalement d'implémenter et d’améliorer les mécanismes complexes d'un site internet interactif : en créant du contenu mis à jour dynamiquement, en appliquant une suite d’instructions ou en exécutant du code selon certains évènements.

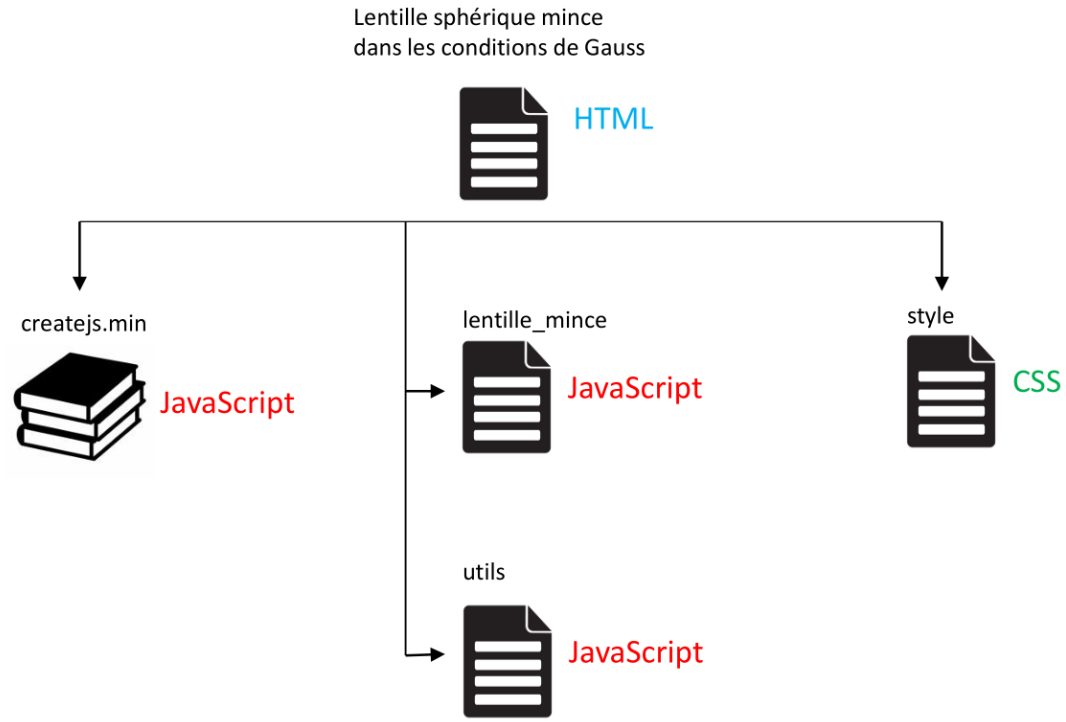


schéma de principe du programme

Le sujet sur lequel nous avons travaillé se base sur un site déjà existant. De ce fait, on a pu récupérer tous les fichiers dont nous avions besoin sur le site : https://www.sciences.univ-nantes.fr/sites/genevieve_tulloue/optiqueGeo/lentilles/lentille_mince.php

Les interactions restent assez basiques, télécharger les fichiers ci-dessus permet de faire fonctionner la lentille. La page HTML intègre les appels aux fichiers CSS et aux 3 JavaScript.

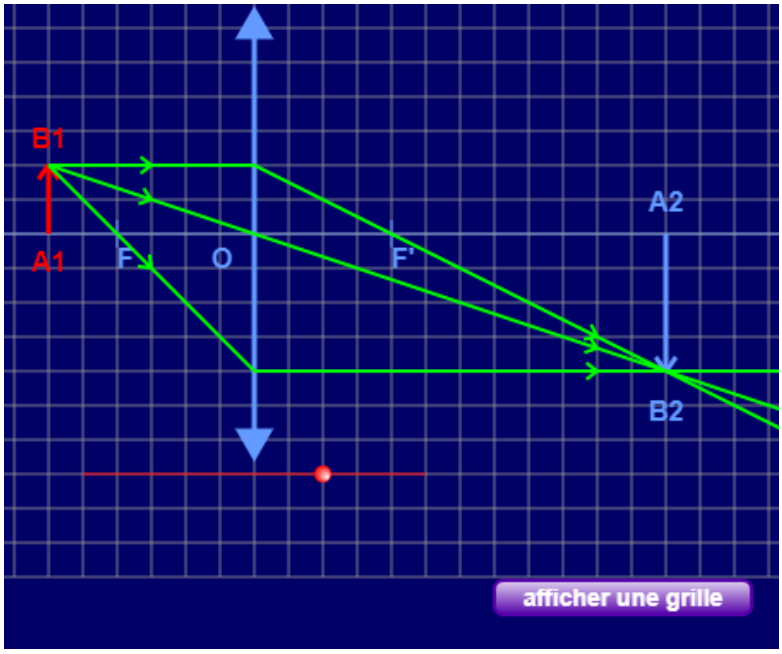
- La bibliothèque createjs.min.js, qui contient tous les objets et les méthodes dont on a besoin pour faire un graphique 2D.

- Le fichier lentille_mince.js qui se sert de la librairie pour dessiner la grande majorité du canevas.

- Le fichier utils.js qui définit le bouton pour passer en plein écran et des surcharges de constructeur.

À l'intérieur de lentille_mince.js, on retrouve une fonction d'initialisation appelée à chaque chargement de la page. Elle regroupe sous forme de conteneur tous les boutons qui apparaissent sur la fenêtre et leurs actions.

```
//grille
var contGrille=new createjs.Container()
var grille=new createjs.Shape()
grille.alpha=0.8;
grille.graphics.beginStroke('#999');
for (var j = -9; j<=10; j++) {
    grille.graphics.moveTo(-400, j*20);
    grille.graphics.lineTo(400, j*20);
}
for (var k = -20; k<=20; k++) {
    grille.graphics.moveTo(-k*20, -180);
    grille.graphics.lineTo(-k*20, 200);
}
contGrille.visible=false;
```



Dans cet exemple se trouve le dessin du quadrillage derrière le bouton « afficher une grille ». Il est créé à partir des objets hérités de CreateJS (Container, Shape) et de leurs méthodes (moveTo, lineTo) qui se servent d'un curseur pour se déplacer dans le canevas et tracer un trait.

Viennent ensuite dans le code les fonctions qui créent le cœur de la page.
Le point B1 représente une source lumineuse qui émet des rayons vers la lentille d'abscisse O. Il peut être déplacé le long des deux axes ce qui influera sur B2. Ce dernier est appelé l'image de l'objet B1 et se situe à l'intersection des rayons émergents. Enfin, le curseur rouge correspond à la distance focale de la lentille. C'est lui qui vient agir virtuellement sur sa forme et déplacer les points F et F'.

Les fonctions majeures dans cette partie sont incident(), emergent() et rayon() qui calculent la trajectoire de la lumière et trace tous les rayons.

La lentille est quant à elle définie comme un objet pouvant se déplacer le long de l'axe x.

II - Le formulaire de paramétrage

Pouvoir bloquer en translation les éléments du graphe a été notre objectif principal durant ce stage. Nous avons convenu tous ensemble dès le départ que la meilleure façon de réaliser ceci était d'avoir un formulaire pour pouvoir saisir à la main les valeurs limites souhaitées. D'autres fonctionnalités sont venues se rajouter au fil des semaines. On y retrouve les champs min et max pour fixer les limites de déplacement en centimètres des 4 protagonistes de la simulation. Ils communiquent avec la partie JavaScript pour faire appliquer les valeurs saisies par l'utilisateur. Il y a aussi la possibilité de bloquer un élément avec Fixe et/ou de le déplacer à un endroit précis avec la dernière balise. Dans les options, on a la possibilité d'afficher une main qui sert de tutoriel et les 4 boutons au pied du canevas qui sont maintenant invisibles par défaut.

Les valeurs affichées correspondent aux bords de la simulation et aux positions initiales.
En appliquant des modifications, on peut visualiser leurs effets dans la foulée.
Le bouton Générer permet de créer et d'enregistrer une lentille avec les mêmes valeurs dans un fichier HTML séparé.

A – Les limites

En se servant des blocs de saisie sur le formulaire ; soit à la main, soit à l'aide de la souris, nous avons tenu compte de la demande de Monsieur Julien Brevier que pour des raisons pédagogiques, les valeurs saisies sur le formulaire qui permettrons une fixation ou des limitations de déplacement doivent être en centimètre, et pas en pixel.
On a pu faire une limitation des valeurs en 2 parties :

- La première partie :

  La limitation lors du remplissage du formulaire qui consiste à ne pas avoir de dépassement de l'intervalle pour chaque valeur minimum et maximum, de telle sorte qu'on ne sorte pas du cadre et qu'on ne dépasse pas le canevas. Une bonne valeur saisie correspond à une animation en vert aux bordures du bloc saisi. Dans le cas contraire, l'animation de la case est de couleur rouge.

- La deuxième partie :

  La limitation après le remplissage du formulaire par des valeurs correspondantes et un clic sur le bouton Appliquer qui appelle la fonction setParametres() et met à jour les valeurs saisies. Chaque modification est écrite dans le corps de la page HTML à l'aide des champs defaultValue et defaultChecked. Ainsi, nous pouvons changer les limites sans jamais avoir à soumettre le formulaire.

Les valeurs saisies sur le formulaire en centimètre, se récupèrent ensuite sur le javascript à partir de leur id. Elles sont traduites en pixels en faisant la conversion avec coef (=0.25) et bornent enfin le déplacement.

```
min <input type="number" id="A1_min" value="-100">
max <input type="number" id="A1_max" value="100">
<input type="checkbox" id="A1_fixe"><label for="A1_fixe">Fixe </label>
<input type="number" id="A1_value" value="-30">
```

déclaration des input dans le formulaire pour A1

```
var A1 = {
    min : Number(document.getElementById("A1_min").value) / coef,
    max : Number(document.getElementById("A1_max").value) / coef,
    fixe : Boolean(document.getElementById("A1_fixe").checked),
    value : Number(document.getElementById("A1_value").value) /coef
};
```

récupération des valeurs dans lentille_mince.js pour A1

La réduction du déplacement d'un élément s'applique en faisant la comparaison entre la position de la souris dans la fenêtre (evt.stageX-depart.x-système.x) par rapport aux limites (ici lentilleHTML.min, max). La fonction ci-dessous est exécutée des dizaines de fois par seconde dès lors qu'on clic et déplace la lentille (pressmove).

Quand on essaye de bouger un objet au-delà de la valeur maximale saisie, chaque position qui sera supérieure à cette valeur va prendre la valeur maximale saisie sur le formulaire. Réciproquement, on mettra à jour les positions inférieures à la valeur saisie sur le formulaire pour qu'elles soient égales au minimum saisi dans le formulaire.

```
lentille.on("pressmove",function(evt) {
    if(lentilleHTML.fixe === false)
    {
        if((evt.stageX-depart.x-systeme.x) >= lentilleHTML.min && (evt.stageX-depart.x-systeme.x)<=lentilleHTML.max)
            evt.currentTarget.x = evt.stageX-depart.x-systeme.x;
        }
        else if((evt.stageX-depart.x-systeme.x) < lentilleHTML.min) {
            evt.currentTarget.x = lentilleHTML.min;
        }
        else if((evt.stageX-depart.x-systeme.x) > lentilleHTML.max) {
            evt.currentTarget.x = lentilleHTML.max;
        }
    }
```

Cette seconde partie de code est exécutée à chaque chargement de la page ou dès qu'on applique le formulaire. Son but est de déplacer la lentille pour qu'elle soit dessinée au bon endroit au cas où, par exemple, sa position par défaut (value) serait inférieure à la limite minimale. La lentille sera déplacée automatiquement à 50 cm si lentilleHTML.min est égal à 50.

```
var lentille=new Lentille(couL,0);
if(lentilleHTML.value > lentilleHTML.max) {
    lentille.x = lentilleHTML.max;
}
else if(lentilleHTML.value < lentilleHTML.min) {
    lentille.x = lentilleHTML.min;
}
else {
    lentille.x = lentilleHTML.value;
}
```

B – Les options

Pour permettre un maximum de fonctionnalités en utilisant cette figure animée, nous devions tenir compte des objectifs des cours et des travaux pratiques que propose Monsieur Brevier à ses étudiants. On nous a demandé de présenter une méthode qui permette d'avoir un contrôle sur l'affichage de quatre boutons. Ils permettent d'afficher les coordonnées des objets ou d'utiliser des modes qui ne sont pas toujours utiles. Nous avons donc trouvé une solution pour pouvoir les faire apparaître à souhait. Le principe pour avoir ce résultat était de mettre ces boutons invisibles par défaut, et c'est au moment de la demande qu'on affiche tel ou tel bouton quand on coche la case appropriée.

Par exemple, pour gérer l'affichage de la main d'aide qui sert de tutoriel, on récupère la valeur de la checkbox afficheMain, puis en fonction de l'état du booléen, on rend visible ou non le conteneur aide qui la contient.

```
var afficheMain = Boolean(document.getElementById("afficheMain").checked);

//aide
var txtEmploi='- le curseur permet de choisir la distance focale de la lent
var aide=new Aide(65,60,txtEmploi);
aide.x=40;
aide.y=40;
if(afficheMain) {
    aide.visible = true;
}
else {
    aide.visible = false;
}
```

C – La génération de fichier

Une des demandes qui nous a été faite par M. Brevier est d'avoir un moyen de sauvegarder les modifications que l'on vient de voir. Le but est d'avoir une lentille pré-paramétrée qui puisse être partagée à ses étudiants pour y travailler dessus. La solution que l'on a retenu pour répondre à cette problématique est d'écrire les valeurs du formulaire dans le corps de la page HTML à chaque fois qu'on les applique. Puis on clone ce fichier et on l'enregistre après avoir masqué le formulaire.

Cette idée rend possible la création d'autant de lentille que l'on veut tout en aillant le même fichier JavaScript. Dans celui-ci tous les champs sont récupérés à partir des attributs des balises puis convertis de centimètres en pixels par coef.

```
var A1 = {
    min : Number(document.getElementById("A1_min").value) / coef,
    max : Number(document.getElementById("A1_max").value) / coef,
    fixe : Boolean(document.getElementById("A1_fixe").checked),
    value : Number(document.getElementById("A1_value").value) /coef
};
```

La fonction generer() exploite ensuite deux notions du javascript : les clones et les Blob. Notre clone est un objet qui vient dupliquer toutes les balises de original.

```
clone de :  ▸<html id="origine">…</html>  type :  object                    lentille_mince.js:846
```

On vient ensuite masquer toute la div qui contient notre formulaire pour pas qu'il n'apparaisse dans la version destinée aux étudiants, puis on transforme notre objet en chaîne de caractères.

```
function generer() {
    var original = document.getElementById("origine");
    var clone = original.cloneNode(true);
    console.log("clone de : ",clone," type : ",typeof(clone));
    var divFormulaire = clone.querySelector("#limites");
    divFormulaire.style.display = "none";
    var cloneTxt = clone.innerHTML;
```

La seconde partie utilise un objet Blob qui prend en paramètres un flux de données plus un type et vient les convertir en une séquence d'octets. Dans notre cas, le stream est la string du corps de la page HTML clonée. Là où ça nous intéresse c'est qu'en créant un objet URL à partir du Blob, on peut synthétiser à la manière d'un hash toute une page dans une quarantaine de caractères.

```
url :  blob:null/d4457c0a-219e-43ef-bd23-e6b467241982
  <a href="blob:null/d4457c0a-219e-43ef-bd23-e6b467241982" download="TP1"></a>
```

On passe ensuite cette URL dans l'attribut href d'une balise « a » et un champ download pour spécifier que l'on souhaite télécharger la cible.
Avec click(), on simule un clic pour lancer le téléchargement du Blob.

```
    var fichier = new Blob([cloneTxt], {type: 'text/html'});
    var url = URL.createObjectURL(fichier);

    var a = document.createElement("a");
    a.href = url;
    a.download = 'TP1';
    document.body.appendChild(a);
    a.click();

    setTimeout(function() {
        document.body.removeChild(a);
        window.URL.revokeObjectURL(url);
    }, 0);
}
```

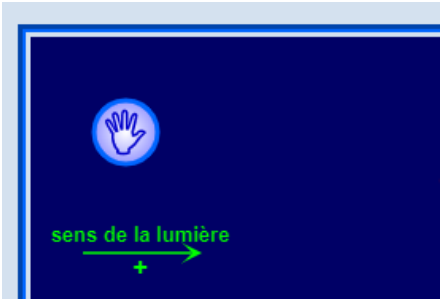
Exemple pour l'affichage de la main d'aide :

Afficher la main d'aide ☑

extrait du formulaire

```
<label for="afficheMain">Afficher la main d'aide</label>
<input type="checkbox" id="afficheMain" checked="">
```

extrait de la string cloneTxt



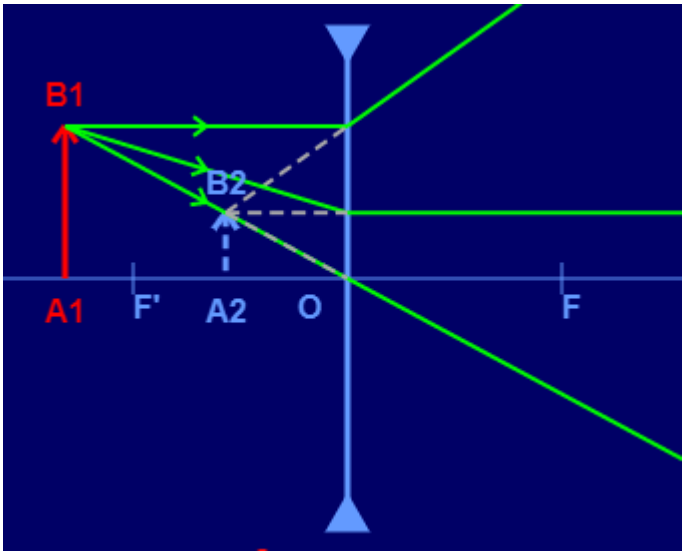
la main d'aide dans l'enfant HTML

III – Les fonctionnalités annexes

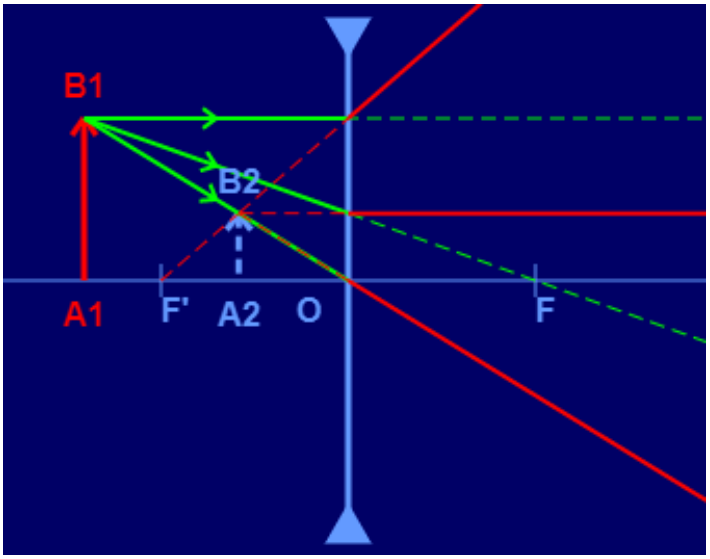
A – Les pointillés

Parmi les ajouts que l'on a fait sur la lentille, les pointillés sont les plus visibles.
Quelques-uns étaient déjà présents dans la version de base. Ils correspondent au prolongement de certains rayons s'ils n'étaient pas déviés ou pour coïncider avec la position d'une image virtuelle. Ils n'existent donc pas réellement mais ils permettent entre autre de comprendre comment trouver la position de ce qui est virtuel.

Ici par exemple, dans le cas où les rayons réfractés sont divergents, on comprend mieux que l'image virtuelle est à leur intersection hypothétique.



avant



après

Dans le code, la fonction calcule() est exécutée des dizaines de fois par seconde dès lors qu'on déplace un élément. Elle se charge d'appeler incident() et emergent() 3 fois chacune.
Elles-mêmes récupèrent la nouvelle position de l'objet qui est en train d'être déplacé et appellent rayon() qui dessine les traits pleins. Une dernière fonction pointille() appelée de façon conjointe à rayon() dessine les pointillés.

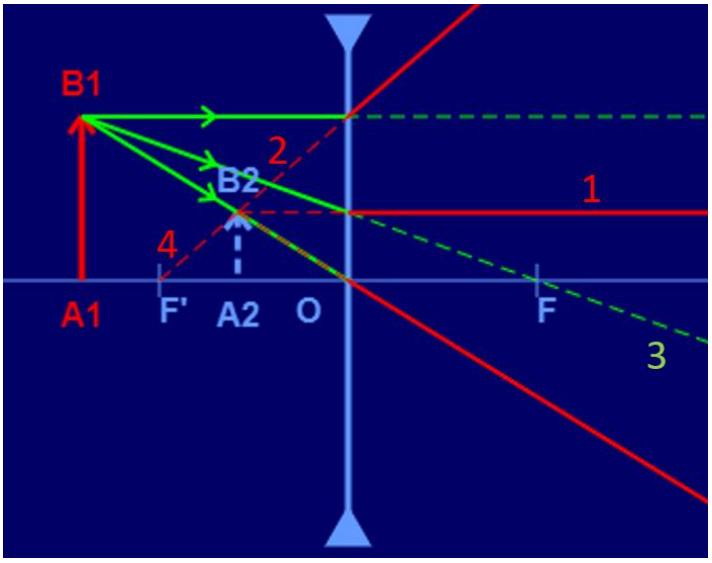
Chaque droite sur le canevas est donc découpée en plusieurs segments de manière logique.
Pour pouvoir les dessiner il faut déterminer où est-ce qu'ils commencent et s'arrêtent.
La lentille venant modifier leur trajectoire, les rayons incidents et émergents doivent être calculés séparément.

Dans le cas où l'image est virtuelle, on dessine en 2 des pointillés partant de P (l'image) et allant vers le x de la lentille (L.x) et le y de l'ordonnée du rayon incident (ordo). Pour le point 3 on part du bord de la fenêtre en calculant un point selon la pente du rayon incident (x = 400, y = yprevu) et on va de nouveau vers la lentille. Le 4 part de Fbis.x et va vers P.x .

Les pointillés pour 3 et 4 mettent en avant le lien entre les distances focales F et F', et la trajectoire des rayons. Cet exemple illustre un des six scénarios qu'il existe dans cette simulation. Dans celui-ci 3 et 4 sont présents car F' est négatif (curFoc.value < 0).

```
function emergent(mc,L,img,ordo, xmin, xmax, ep, f){
    var P=new createjs.Point(img.X,img.Y);
    var pente = (img.Y-ordo)/(img.X-L.x);
    var yfin=ordo+(xmax-L.x)*pente;
    var ydeb=ordo+(xmin-L.x)*pente;
    var inci=new createjs.Point(L.x,ordo);
    var arriv=new createjs.Point(xmax,yfin);
    var penteinitiale=(ob.Y-ordo)/(ob.X-L.x);
    var yprevu=ordo+(xmax-L.x)*penteinitiale;
    var Fbis = new createjs.Point(curFoc.value + lentille.x, 0);

    mc.graphics.setStrokeStyle(ep).beginStroke(couEmergent);
    if (img.reel) {
        //image réelle
        rayon(mc,inci, arriv, f);
    }
    else{
        //image virtuelle
        mc.graphics.beginStroke('#AAA');
        pointille(mc,P,inci,couEmergent);                                   2
        if(curFoc.value <= 0) {
            pointille(mc,new createjs.Point(xmax,yprevu), inci, couIncident);   3
            if(ordo == ob.Y) {
                pointille(mc,Fbis,P,couEmergent);                           4
            }
        }
        mc.graphics.setStrokeStyle(ep).beginStroke(couEmergent);
        rayon(mc,inci, arriv, f);                                           1
    }
}
```



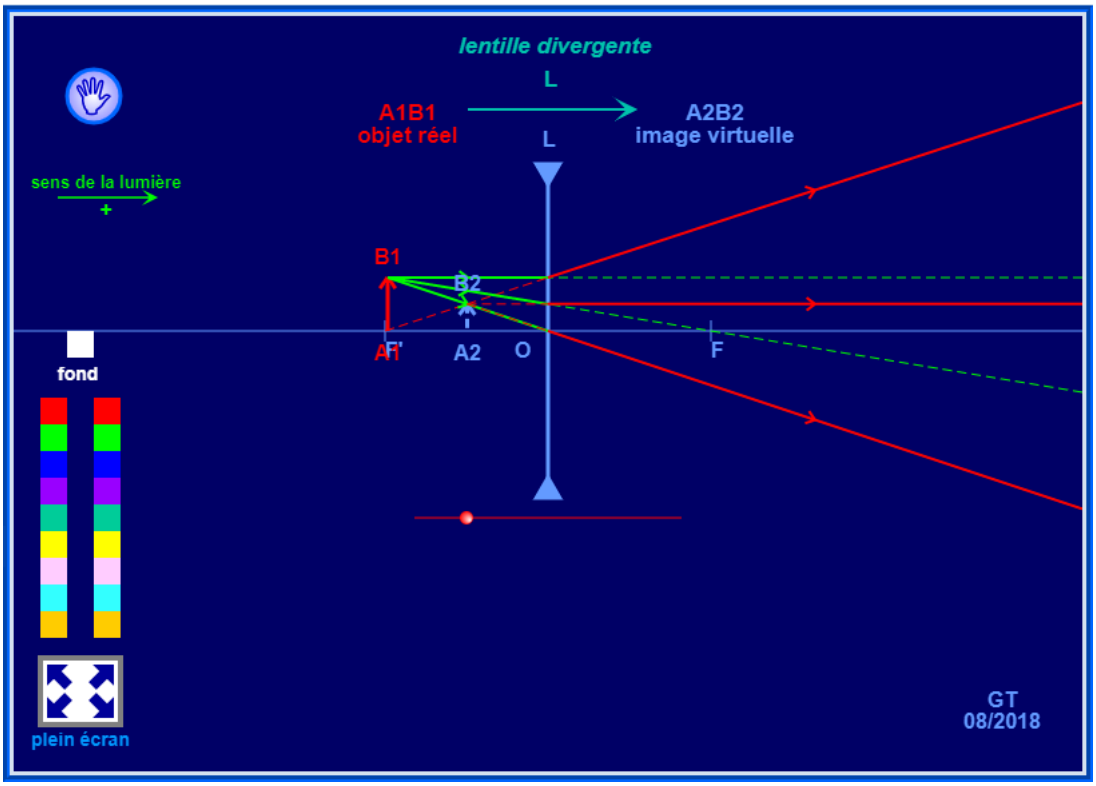
B – Les palettes de couleurs

Comme on est sur une véritable station de travail graphique, qui regroupe des rayons lumineux incidents et émergents, une lentille, un objet et une image ; la couleur des rayons incidents et émis à travers la lentille étaient les mêmes et peuvent être changés en fonction de neuf couleurs proposées. Graphiquement c'est trop bien cette harmonie de couleur : même couleur pour tous les rayons. Mais au fond, elle présente un problème majeur.
À titre d'exemple, lorsqu'on passe d'un objet réel à virtuel, on ne comprend pas bien où sont les rayons incidents et émergents.

Suite à ce problème, il nous a été demandé de créer une nouvelle tablette de couleurs de sorte que celle de gauche permette de retoucher la couleur des rayons incidents, et la deuxième ajuste la couleur des rayons émergents.

Techniquement, on s'est servi du code de la première palette de couleurs pour en créer une deuxième, tout simplement en créant un nouveau conteneur qui présente la même liste de couleurs sur une autre position.

On a dupliqué par la suite une variable cou qui était responsable d'ajuster les couleurs des rayons incidents et émergent à la fois. On a eu ensuite deux variables couIncident et couEmergent qui ont pris le rôle de déterminer les couleurs des rayons et leurs déviations.



IV – Bilan

Finalement, nous avons réussi à mener à bien ce projet dans le temps qui nous était imparti. Même si d'autres ajouts auraient été souhaités, l'essentiel de ce qui nous était demandé a été fait. On a su implémenter nombre de petites fonctionnalités en parallèle de l'idée de base. Les réunions fixées chaque semaine nous ont permis d'échanger avec Mr Brevier et d'avoir un retour sur ce que l'on faisait. On a pu avancer dans la bonne direction de cette façon et s'adapter au besoin. En comprenant de mieux en mieux le code existant, les attentes paraissaient de plus en plus atteignables et le côté relation client est venu renforcer notre motivation pour y arriver.

En plus d'avoir découvert ce qu'est la relation client dans un projet concret de programmation, ce stage nous a donné l'occasion de mettre en pratique les notions que l'on a vu en développement web et en informatique graphique.

La bibliothèque de CreateJS définit des objets et des méthodes qui ressemblent fortement à ceux qu'on a utilisé avec Processing. Le fait d'avoir déjà eu à manipuler des formes géométriques dans un espace en 2D nous a aidé dans la compréhension du code existant. Ce stage nous a donné en plus l'opportunité de mettre à l'ouvrage et d'accroître nos connaissances en programmation graphique au fil des semaines.

Quant au développement web, nous en avons grandement appris ces deux derniers mois. Les apports que l'on en retire s'étalent sur une partie essentielle du front-end. Il existe sans doute encore énormément de choses à apprendre des sites internet mais la compréhension que l'on en a s'est largement améliorée grâce à ce projet. Le trio HTML, CSS, JavaScript est le pilier de cette discipline et nous sommes ravis d'avoir pu consolider nos acquis dans ce domaine. En travaillant sur ces bases, nous comprenons maintenant mieux comment fonctionnent les formulaires, les interactions JavaScript - HTML et la syntaxe du JavaScript dans sa globalité.

Nous avons donc pu nous servir des cours de cette année et les approfondir tout en ayant la satisfaction d'avoir pu utiliser ces connaissances dans quelque chose d'utile à quelqu'un. Nous sommes confiants quant à l'idée d'avoir à travailler de nouveau sur un site et sur le fait que nous avons dorénavant tous les éléments qu'il faut pour réussir.