Patates parallèles

Vincent Lesbros

14/5/2021 - 18/5/2021

Une ‘Patate’ est une chaîne de couplages élastiques circulaire de dimension D.

Les maillons de la chaîne maintiennent une position et une vitesse pour chaque dimension.

Un cycle consiste en le calcul des forces appliquées sur chaque maillon par ses maillons voisins puis la conversion de ces forces en accélérations pour calculer les nouvelles positions et vitesses. C’est un cas particulier de réseau de couplages élastiques. Les tensions des couplages peuvent être données localement à chaque maillon tension t_p et t_s avec le maillon précédent et le maillon suivant, soit uniformément pour toute la chaîne. On peut ajouter une tension vers 0 pour chaque maillon.

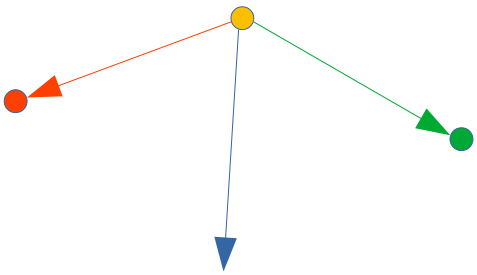
La synthèse sonore correspondante consiste simplement à échantillonner les positions des maillons. Soit directement pour obtenir le son fondamental correspondant au périmètre de la chaîne, soit en ré-échantillonnant pour modifier la fréquence.

Le périmètre est le nombre de maillons. La fréquence fondamentale f est donc la fréquence d’échantillonnage F_e divisée par le périmètre. Le périmètre P est la période du fondamental donnée en nombre d’échantillons. La fréquence fondamentale est $f = F_e / P$.

Les sons produits sont donc tous issus de périodes entières et sont des sons harmoniques, mais le jeu sur les tensions permet, en modifiant les phases, de générer des fréquences correspondant à des périodes non entières par une circulation des valeurs de position le long du périmètre au cours du temps, à l’échelle de la période.

Modèle parallèle de dimension 2

Notre premier modèle sera de dimension 2 pour générer des sons stéréo.



Sur chaque maillon (orange) on applique en parallèle trois forces : en rouge la force vers le maillon précédent, en vert la force vers le maillon suivant et en bleu la force vers zéro. Un coefficient pour chacune permet de moduler indépendamment les trois forces.

Un coefficient de fluidité permet de réduire globalement la vitesse des maillons après l’application des forces et avant l’application des vitesses aux positions.

Ce n’est pas un modèle physique : si les tensions précédentes et suivantes ne sont pas égales, le système peut gagner de l’énergie et même exploser. La tension vers zéro permet de rétablir les positions dans l’intervalle [-1, 1] compatible avec les amplitudes des échantillons sonores.

Création d’une patate

- Soit on spécifie un périmètre et on rempli aléatoirement les positions avec des valeurs de l’intervalle [-1, 1]. Les vitesses initiales sont à 0. On peut remplir les positions initiales avec un signal particulier, sinusoïdal ou impulsion par exemple.

- Soit on enregistre les mouvements de la souris au cours d’un clic, on normalise les valeurs entre -1 et 1 (non implémenté dans cette version).

Représentation dans les textures pour le calcul parallèle

Pour commencer, on donnera les tensions dans des uniformes. Les textures de patate ne contiendront donc que les positions et vitesses pour deux dimensions.

T0/T1

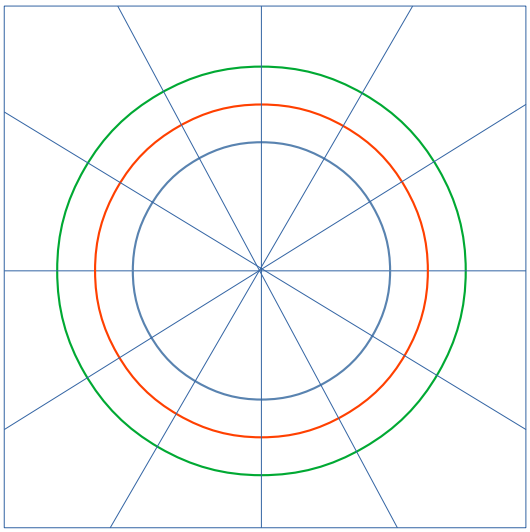
p_x	v_x
p_y	v_y

Un pixel contient 4 valeurs (RGBA). Les textures T0 et T1 contiendront les positions et vitesses en x et en y. On alternera entre les deux textures d’un cycle au suivant, ou on recopiera T1 dans T0 après le calcul d’un cycle.

La texture peut être de H pixels de haut et P pixels de large pour représenter H patates de périmètre P. Pour cette version, H = 1.

Visualisation

On souhaite représenter une dimension de la patate en deux dimensions sur l’écran dans un cercle inscrit dans un carré :



L’angle représente la phase, c’est-à-dire un indice dans la période ou périmètre. Le cercle rouge représente la position 0, le cercle vert la position 1 et le cercle bleu la position -1.

Pour chaque point A de l'image à générer, on calcule dans un repère centré l'angle Θ en radians et la distance au centre *d*. L'angle est traduit en index dans la texture de patate :

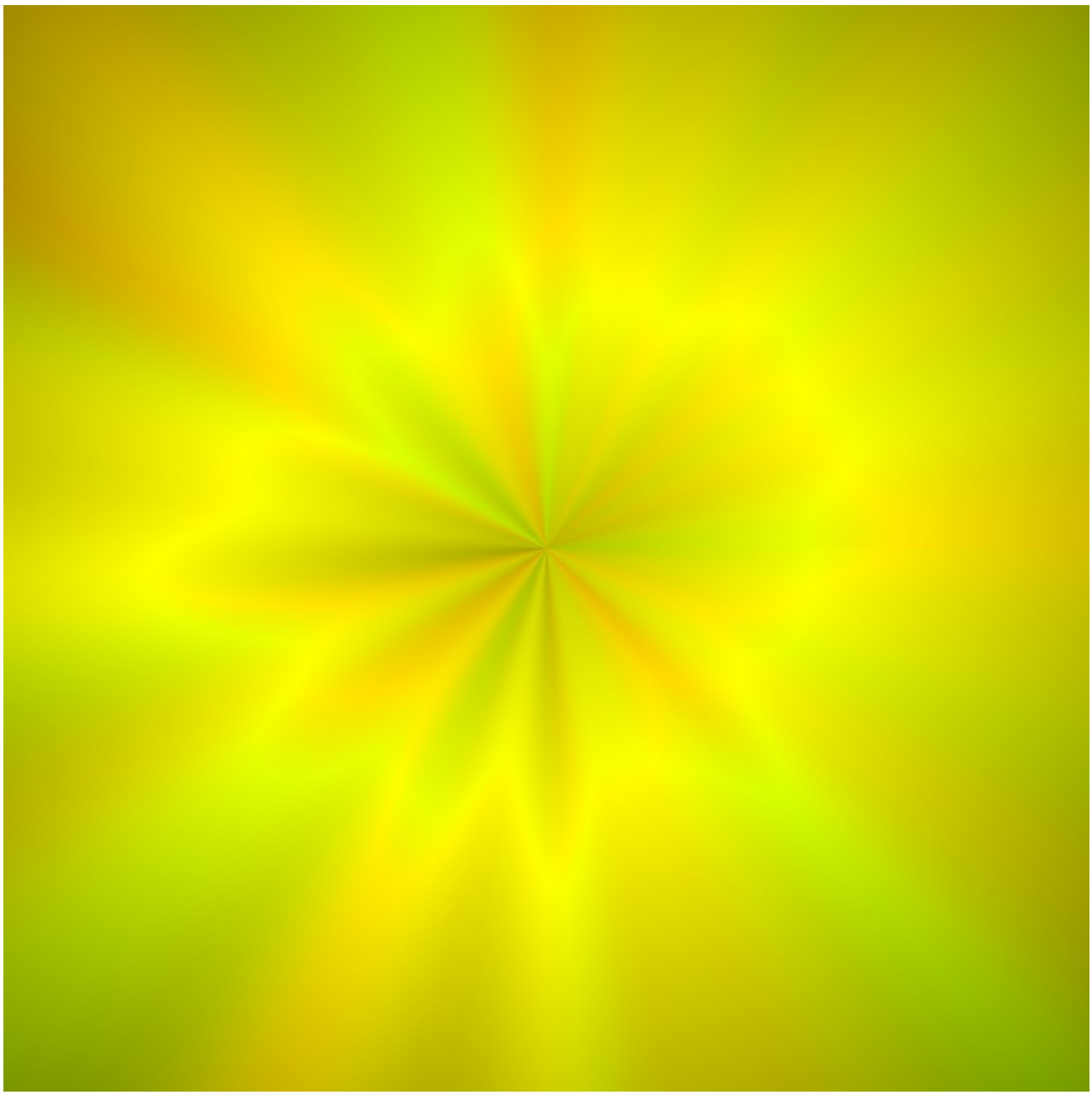
$x = périmètre * \Theta / 2\pi$.

Ce *x* associé au *y* correspondant à la patate permet de trouver la position *p* de la patate pour cette phase. On détermine ensuite une règle de transformation de la différence entre *p* et *d* en couleur en prenant en compte les rayons attribués au 0, au -1 et au 1. Par exemple plus lumineux si la traduction de *d* en position d'après les rayons des cercles correspond à la position *p*.

Plus exactement, les *texels* doivent viser le milieu des pixels pour obtenir une valeur exacte. Les centres des pixels pour un périmètre p varient de 0.5 à p - 0.5 (en pixels), donc de (0.5 / p) à (p - 0.5) / p.

D'où la formule finale :

```
highp float x = (((u_taille_0.x - 1.0) * theta) / (MathPI * 2.0) + 0.5) / u_taille_0.x;
```



Le rayon r_0 est 1/4, le rouge et le vert correspondent aux deux dimensions.

Calcul du ré-échantillonnage d'une période

L'instruction '`i21519`' prend en entrée la texture T0 de la patate à deux dimensions, une ligne de pixels $\{p_x, v_x, p_y, v_y\}$ et en sortie la texture T2 de la forme d'un signal stéréo $\{g_0, d_0, g_1, d_1\}$.

T0/T1

p_x	v_x
p_y	v_y

T2

g_0	d_0
g_1	d_1

Pour le premier échantillon des deux, on calcule la phase du *texel* courant `phase` en fonction de la taille de la texture de sortie `u_resolution`. On reporte la phase dans la texture d'entrée de taille `u_taille_0` pour calculer x, en *texel*.

Pour le second échantillon, la phase est calculée un demi pixel plus loin.

Cela n'est valable que si la période en nombre d'échantillons est inférieure à la taille maximale d'une texture, on table sur 16384 a priori mais on interroge WebGL pour le déterminer, et que la période est un nombre pair.

```
// premier échantillon
highp float phase = (v_texel.x * u_resolution.x - 0.5) * MathPI * 2.0 / u_resolution.x;
highp float x = (((u_taille_0.x - 1.0) * phase) / (MathPI * 2.0) + 0.5) / u_taille_0.x;

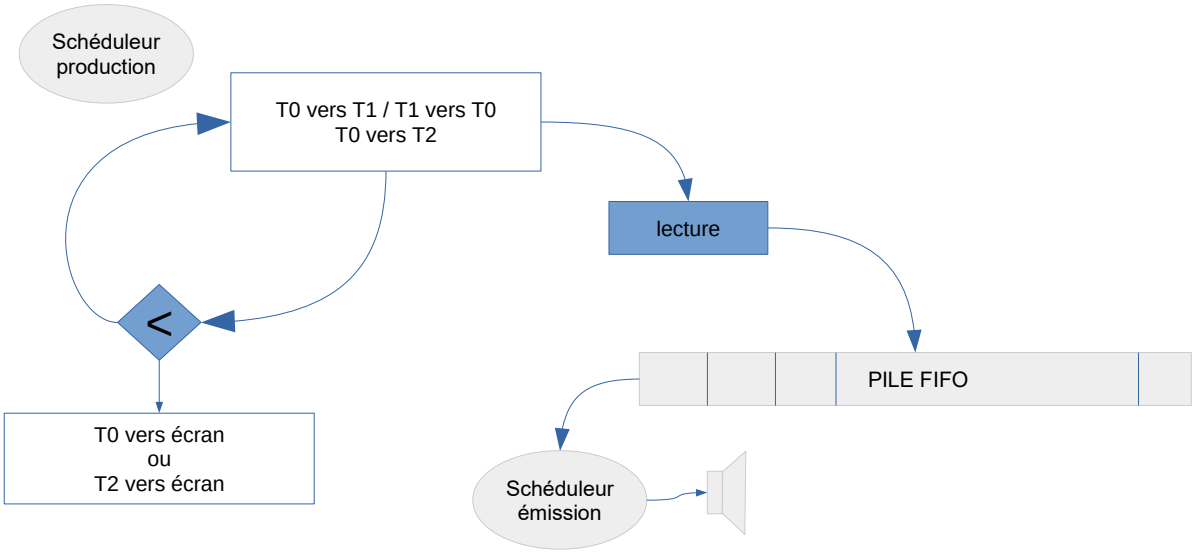
highp vec2 t0 = texture2D(u_entree_0, vec2(x, 0.5)).xz; // on ignore les vitesses
// position et vitesse du point
highp float g0 = t0.x; // gauche
highp float d0 = t0.y; // droite

// second échantillon
phase = (v_texel.x * u_resolution.x) * MathPI * 2.0 / u_resolution.x; // 1/2 plus loin
x = (((u_taille_0.x - 1.0) * phase) / (MathPI * 2.0) + 0.5) / u_taille_0.x;
t0 = texture2D(u_entree_0, vec2(x, 0.5)).xz; // on ignore les vitesses
highp float g1 = t0.x; // gauche
highp float d1 = t0.y; // droite

// sortie deux couples G,D
gl_FragColor = vec4(g0, d0, g1, d1);
```

Schéma du programme

Le programme fonctionne avec deux « *schéduleurs* ». Un schéduleur de production et d'affichage, et un schéduleur de consommation et d'émission du son.



La production empile des périodes de son dans un *buffer* tant que le temps de calcul ne dépasse pas une limite, fixée à 20 millisecondes actuellement. Dès que ce temps est atteint ou dépassé, la texture de position/vitesse T0 ou la texture de période T2 est traduite en image et affichée. Les rappels à la boucle de production sont gérés par '*request frame*' du navigateur. Tous les calculs de cette boucle sont envoyés au GPU par l'intermédiaire de l'API WebGL 1, avec quelques extensions comme la possibilité de traiter des textures en nombre flottants et en interpolation linéaire.

La fin d'un cycle (rectangle 'lecture' en bleu) est un transfert GPU vers CPU pour lire les nouveaux échantillons d'une période, qui sont recopiés et empilés à destination du consommateur. Lors de la lecture, plusieurs périodes de son peuvent être cumulées dans un même *buffer* pour limiter l'*overhead*. Dans l'étude des temps de calculs, c'est ce transfert qui prend le plus de temps.

Le consommateur est une boucle opérant avec un *timer* réglé à 25 millisecondes. Tant qu'il y a des *buffers* remplis dans la pile, le 'sablier' construit une source audio, calcule le moment de l'émission et émet le son avec une fonction *callback* pour libérer le *buffer* dès que le jeu est terminé.

On a réglé à une demie seconde le retard du son sur l'image. Comme lorsqu'on regarde et écoute un orage, le son arrive après l'éclair ! Si jamais le schéduleur de consommation arrive en retard, la boucle est arrêté avec un message indiquant le nombre de seconde de retard. On peut alors changer les réglages et relancer.

Conclusion en résumé

L'idée est peut être complètement folle : on part d'une représentation des sons qui sont des ondes sphériques longitudinales se propageant dans un milieu, air ou eau par exemple, sous forme d'une table numérique des valeurs d'amplitude échantillonnées dans le temps. C'est ce qui est utilisé couramment pour reproduire les sons, en mp3 après compression ou avec des disques vinyles de manière analogique sans échantillonnage.

On considère que le son est périodique et on étudie une période.

On ‘matérialise’ ces valeurs instantanées d’amplitude et on leur attribue des comportements : on relie les échantillons par une simulation d’élastiques. À chaque cycle de calcul, on produit une nouvelle période du signal en cours de génération, on traduit cette période en image et en son.

Perspectives

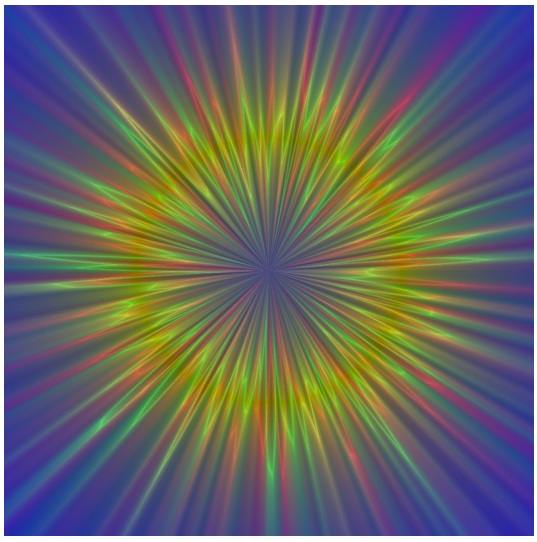
Ajouter des ‘masses’ au maillons, individualiser les maillons avec des tensions locales à la place ou en plus des tensions uniformes.

Permettre de modifier le voisinage en définissant les décalages à utiliser pour les voisins précédents et suivant.

Ajouter une régulation automatique pour empêcher les explosions : par exemple en modifiant la tension vers 0 en fonction de seuil sur les amplitudes.

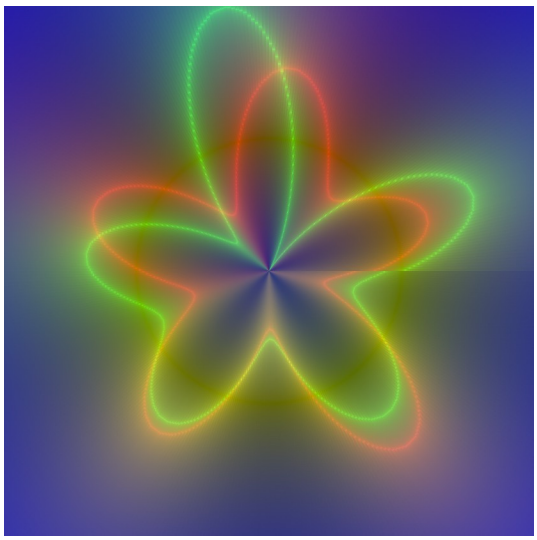
Créer de nouvelles façons d’exciter un système à l’équilibre. Pour l’instant, soit tout aléatoire soit une impulsion soit un ‘sinus/cosinus’. Il manque l’équivalent d’un archet.

Mode d’emploi

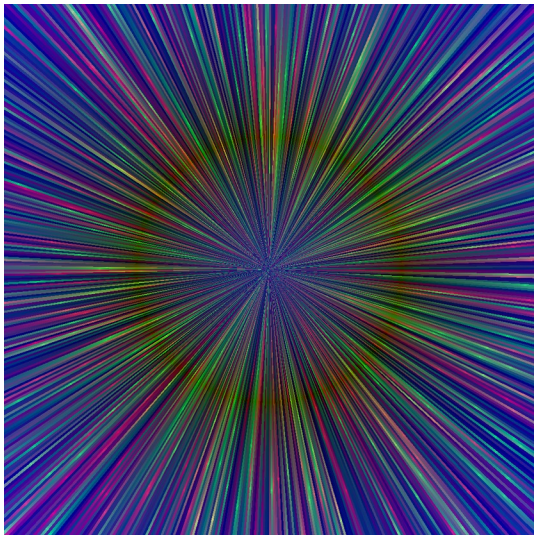


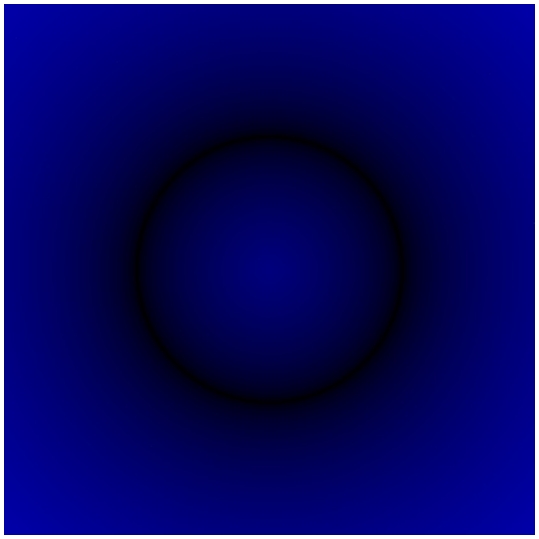
Ouvrir la page : https://www.cyclonium.com/atelier/couplages/patatesParallelles.html

Appuyer sur ‘Démarrer’, puis régler les paramètres : tensions et périodes. Le son doit être produit environ une seconde après le démarrage.

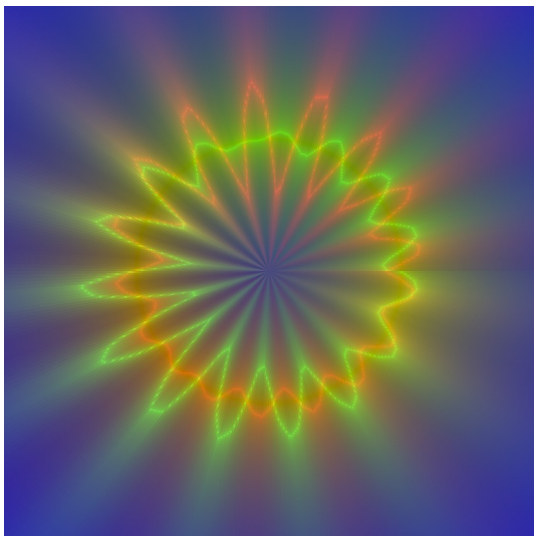


En rouge l’image du canal gauche et en vert celle du canal droit.





Une explosion débutante à gauche et terminée à droite. Dans ce cas, on peut augmenter la tension vers zéro pour rattraper le coup.



La case à cocher ‘Affichage patate ré-échantillonnée / fondamentale’ permet de choisir entre l’affichage par défaut de la patate ré-échantillonnée, ou la patate fondamentale utilisée dans le calcul. Cela ne change pas le son mais uniquement l’affichage.

Note : les périodes dans l’interface sont affichées en valeurs flottantes. En fait, elle sont arrondies à l’entier pair le plus proche.