TP Approfondissement GPU HeatTransfert

Thibaut Piquerez

28.01.2019

TP choisit

HeatTransfert

- Simulation de transfert de chaleur

TABLE DES MATIÈRES

Interactions

01	Mettre le crayon à température froide	Touche C
02	Mettre le crayon à température chaude	Touche H
03	Activer la gomme pour effacer le “Heater”	Touche R
04	Utiliser le crayon ou le gomme	Clique gauche
05	Changement de la taille du crayon	Molette
06	Changement de “Heater” persistant à non-persistant	Touche P

Implémentations réalisées

- Cuda
  - avec Global Memory
  - avec Textures
- OpenMP
  - forAuto
  - entrelacement

Structure du code

- 3 kernels pour la version global memory
- 3 kernels pour la version texture
- 1 classe côté host
  - Appel les kernels ci-dessus en fonction de la version
- Gestion des événements
  - Appel des fonctions de classe côté host

Persistance des “Heaters”

- Booléen
- Si persistant
  - L’image contenant les “Heaters” écrase l’image affichée à chaque itération
- Si non-persistant
  - L’image contenant les “Heaters” écrase l’image affichée uniquement lors d’un ajout
  - Réinitialisation des “Heaters” après écrasement

```
diffusion<<<dg,db>>>(ptrImageInput, ptrImageOutput, w, h);
if (heaterPersistant || ecrasementFlag)
      {
      ecrasement<<<dg,db>>>(ptrDevImageHeater, ptrImageOutput, w, h);
      }
```

```
if (ecrasementFlag && !heaterPersistant)
      {
      ecrasementFlag = false;
      resetHeaters();
      }
```

Textures binding

```
texture<float,2,cudaReadModeElementType> textureHeater;
```

```
__host__ void initTextureHeater(float* ptrDevHeater, int w, int h)
   {
   textureHeater.addressMode[0]=cudaAddressModeClamp;
   textureHeater.addressMode[1]=cudaAddressModeClamp;
   textureHeater.filterMode=cudaFilterModePoint;
   textureHeater.normalized=false;

   size_t pitch = w * sizeof(float);
   cudaChannelFormatDesc channelDesc = cudaCreateChannelDesc<float>();
   HANDLE_ERROR(cudaBindTexture2D(NULL,textureHeater,ptrDevHeater,channelDesc,w,h,pitch));
   }
```

Texture vs GM

Version Global Memory

```
if(ptrDevHeater[s]>0.0)
	{
	ptrDevOutput[s]=ptrDevHeater[s];
	}
```

Version Textures

```
if(tex2D(textureHeater,j,i)>0.0)
	{
	ptrDevOutput[s]=tex2D(textureHeater,j,i);
	}
```

Un booléen permet de switcher entre la version Texture et GM

Texture - Diffusion et conversion

- Possède une texture
- Binding à chaque fois avec la bonne image (Image A ou B)

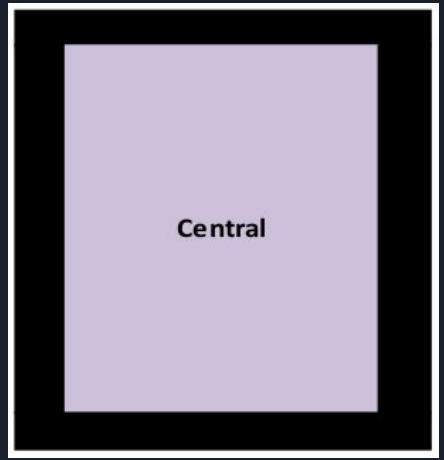
```
initTextureAB(ptrImageInput, w, h);
diffusionTex<<<dg,db>>>(ptrImageOutput, w, h);
```

```
initTextureABHSB(ptrImageOutput, w, h);
toScreenImageHSBTex<<<dg,db>>>(ptrDevPixels, w, h, *calibreur);
```

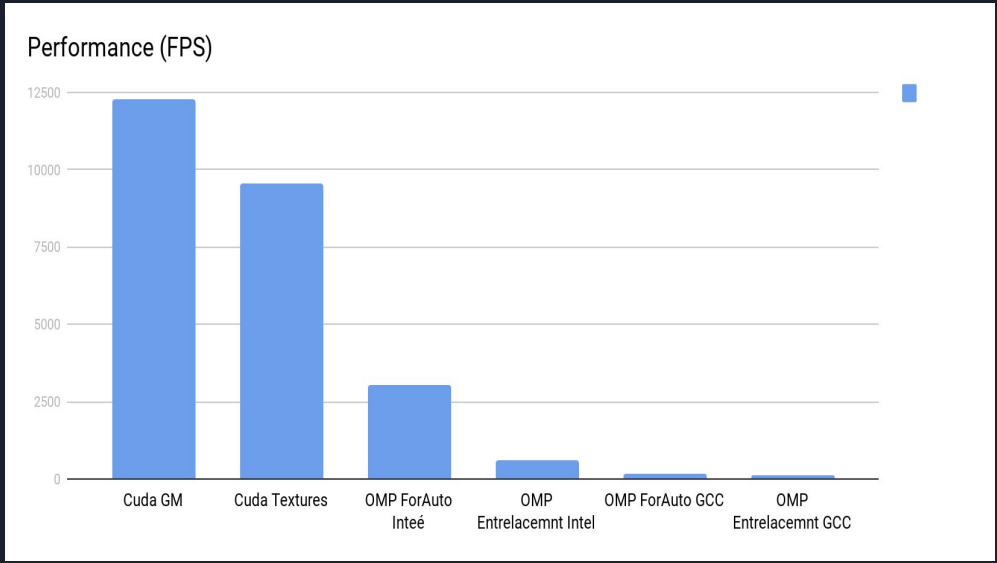
Suppression d’un “if” lors de la diffusion

```
const int TID = Indice2D::tid();
const int NB_THREAD = Indice2D::nbThread();
const int WH = (w-2) * (h-2);
int s = TID;
int i;
int j;

while (s < WH)
      {
      IndiceTools::toIJ(s, w-2, &i, &j);
      int sReal = s+w+2*i+1;

      ptrDevOutput[sReal]=ptrDevInput[sReal]+0.25*(ptrDevInput[sRe
      al+1]+ptrDevInput[sReal-1]+ptrDevInput[sReal+w]+ptrDevInput[
      sReal-w]-4*ptrDevInput[sReal]);

      s += NB_THREAD;
      }
```

- 2 pixels de moins en largeur et hauteur
- Calculer WH sans ces pixels
- Calculer le “s” réel



Comparaison des performances



Implémentation	FPS
Cuda GM	12261
Cuda Texture	9538
OMP ForAuto Intel	3025
OMP Entrelacement Intel	604
OMP ForAuto GCC	144
OMP Entrelacement GCC	106

Démo

Améliorations

- Binding des textures en boucle pour diffusion et conversion
- Ajout d’éléments visuels
  - Affichage de la taille du crayon ou de la gomme
  - Etat du crayon (froid, chaud, gomme)
- “Heaters dynamique”
  - Ajouter une forme qui émet de la chaleur pendant un certain temps
- Enregistrement des “heaters”

Conclusion

- TP fonctionnel avec 4 versions
- Grande différence entre OpenMP et Cuda
  - OpenMP ForAuto avec Intel pas mauvais non-plus

Questions ?