

# Synthèse d'images

TD4



# Moteur 3D

# Contenu d'un moteur 3D

- Très variable selon les fabricants
  - Obligatoirement :
    - Couche haut niveau d'accès à l'affichage, aux GPUs
    - Performance temps réel
  - En option :
    - Chargement des données graphiques à partir de données
    - Moteur d'éclairage, de physique, son, particules, animations, cycle jour/nuit, ciel, nuage, mer/océan, météo, etc.
    - Éditeur de scène / de niveaux / de cinématiques
    - Scripting
    - Modelling
    - Exporteur/plugin depuis un modelleur
    - Etc.

# Objectif d'un moteur 3D

- Simplifier le travail des développeurs
  - En fournissant des fonctionnalités clé-en-main
  - Systèmes (hardware) de plus en plus complexes
  - Innovations et algorithmes (software) de plus en plus pointus
  - Concentration de l'expertise dans un produit intermédiaire (middleware)
- Mutualisation des coûts
  - Économie d'échelle
  - Concentration du marché

# Exemples de fonctionnalités

- Géométries, maillages (TD2)
- Matériaux, shaders (TD3)
- Brefs aperçus (TD4)
  - Scenegraph/structure d'accélération
  - Shadow map
  - Moteur d'animation
  - Moteur de particules
  - Moteur de physique
- Non abordés
  - Météo/environnement, GUI, son, réseaux, scripting, édition, m...

# Scénographie

# Organiser sa scène

- Relation entre les objets
  - Le stylo au bout des doigts
  - Les roues d'une voiture
  - La chaise sur le sol
- Les objets s'organisent les uns par rapport aux autres

# Structurer ses données

- Data-driven
- Bon pour le cerveau
  - Permet de s'y retrouver
- Bon pour le CPU
  - Permet d'optimiser les performances
- Bon pour le GPU
  - Permet d'optimiser les performances



# Structurer ses données

- Tuples
- Tableaux (1 à n dimensions)
- Listes
- Ensembles
- Dictionnaires (map)
- Arbres
- Graphes
- Etc.

# Graphe de scène

# Graphe de scène (*scene graph*)

- « Structure » habituellement usitée dans la manipulation de scène 3D
- Mais pas de réelle définition !
  - Un tableau ou une liste est un graphe de scène
- Vous pouvez jeter toute la littérature dessus
  - Chacun a sa propre définition, sa propre interprétation et sa propre implémentation

# Exemples d'implémentation

- OpenSceneGraph
- OpenSG
- Middleware (moteurs 2D/3D, de jeu)
- Three.js Object3D

# Que retenir ?

- C'est un graphe
  - Constitué de nœuds (*node*)
  - Généralement acyclique
  - Souvent arborescent (*tree*) : relation de parenté
  - Logique d'association : 2 nœuds du graphe qui sont connectés ont une relation particulière
- Il gère le contenu de la scène
  - Le contenu est représenté par les nœuds du graphe
- C'est une sorte de patron de conception (*design pattern*)

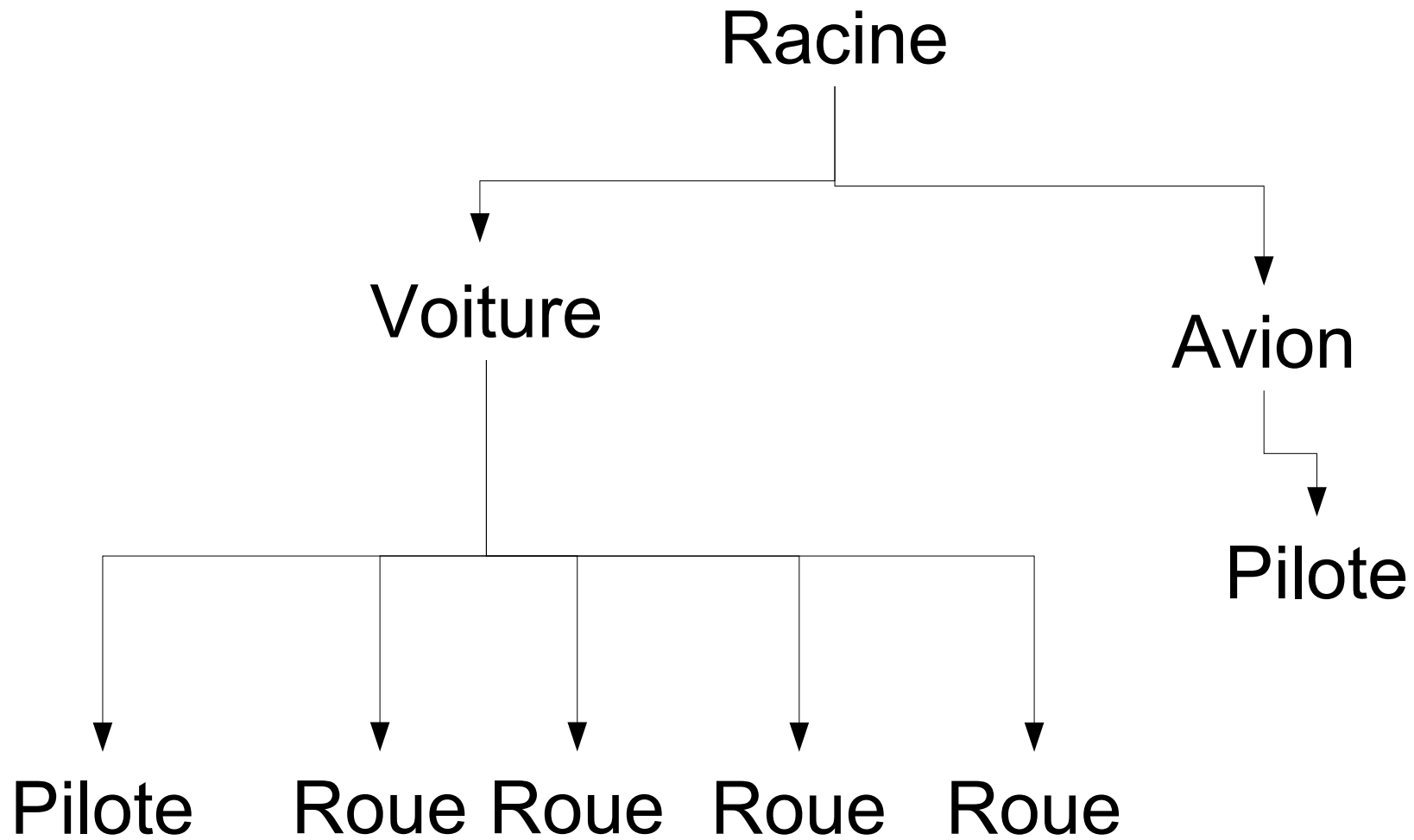
# Architecture habituelle

- La relation de parenté est associée à la spatialité
    - Les *nodes roues* (d'une voiture) seront enfants du *node chassis* (de la voiture)
  - La structure du graphe est dynamique
    - Le stylo sur la table change de *parent* quand il est pris en main
  - Le *design pattern Traversal* permet de parcourir le graphe
    - Le graphe n'est que les données
    - Les opérations fonctionnelles sont exécutées par les objets
- Traversal :*
- ex : (*query*) récupérer la liste des nœuds de type *chassis*

# Limitations

- Architecture lourde
  - Les objets *node* sont des « gros » objets informatiquement parlant
  - Pas adapté pour les performances brutes
- Le graphe acyclique ne gère pas tous les cas
  - Le stylo sur la table qu'on pousse avec un doigt : est-il le nœud fils de la table (association par la gravité) ou du doigt (actionneur du mouvement) ?
  - Le projectile est-il associé au tireur, ou à la cible, ou indépendant ?
- Le *design pattern Traversal* est pénible
  - Un peu comme le *pattern Visitor*

# Exemple

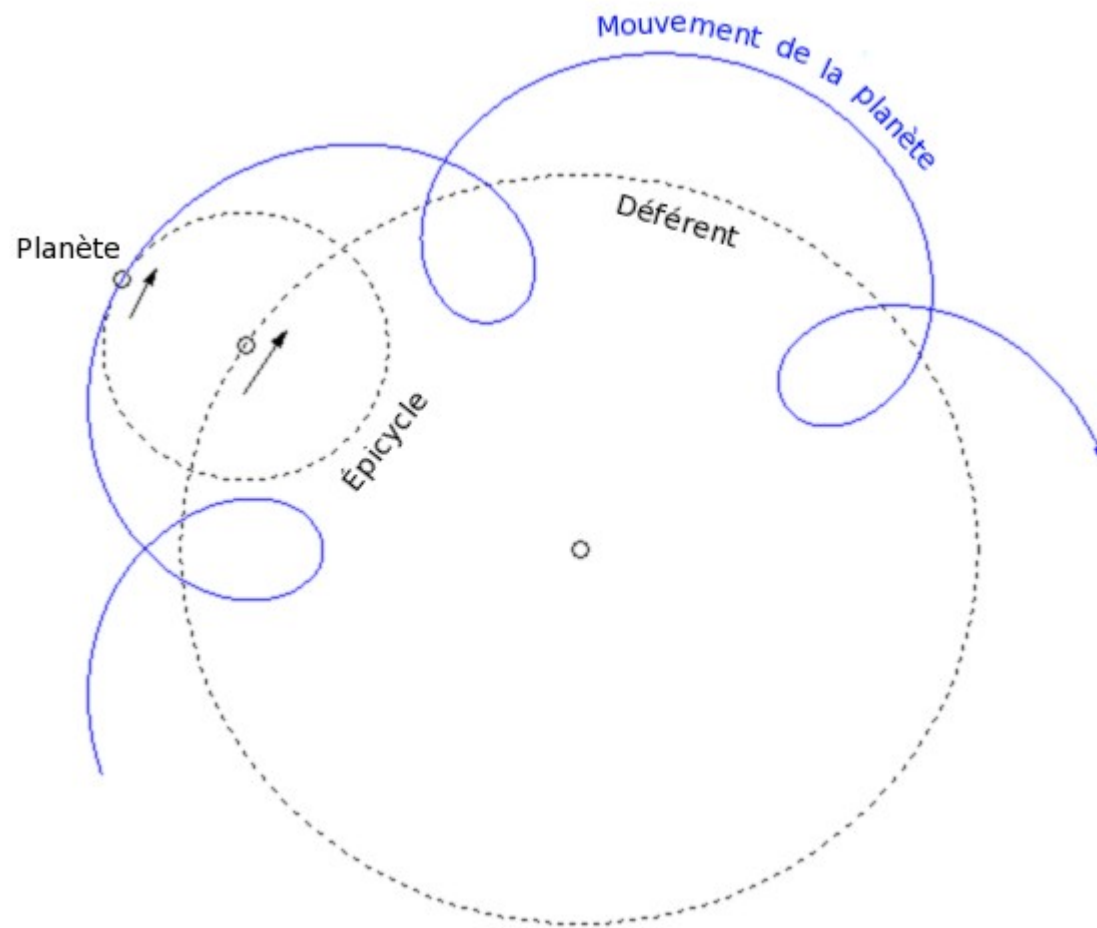




# Chaînage des transformations

- La représentation en arbre se prête bien à la cumulation de transformations
  - L'objectif est au final de connaître les positions par rapport à la caméra
  - La position de certains objets peut être complexe dans un repère absolu
    - Comment calculer la position de la Lune par rapport au Soleil ?
  - Mais simple dans d'autres repères
    - Lune = Trajectoire circulaire autour de la Terre
    - Terre = Trajectoire circulaire autour du Soleil

# Chaînage des transformations



# Chaînage des transformations (suite)

- Outil mathématique : la matrice
  - Chaîner 2 transformations revient à une multiplication de matrices
    - C'est super cool
    - C'est ce qu'on a déjà fait dans le vertex shader
- Généralement implémenté dans les *scene graph*
  - Le calcul de position se fait automatiquement du moment que les objets sont dans un *scene graph* avec une relation parent/enfant
  - C'est le cas dans Three.js

# Structure d'accélération

# Problématique

- Une scène peut être très conséquente
  - Ex : Open world
- Selon le traitement effectué la quantité d'objets à traiter peut être un goulot de performance
  - Affichage
  - Calcul de collisions
  - Mise à jour de l'état des agents/acteurs
  - Etc.

# Solution

- Ne traiter qu'une partie des objets !
- Comment réduire efficacement le champ des objets à traiter ?
- Les structures d'accélération permettent cela dans le cas des traitements relatifs à l'organisation spatiale

# Confusion courante

- Dans beaucoup de moteurs, les notions de graphe de scène et de structure d'accélération se superposent bien que les 2 concepts répondent à 2 problématiques séparées
  - Graphe de scène : problématique d'organisation des données
  - Structure d'accélération : problématique de performance

# Exemple d'usage

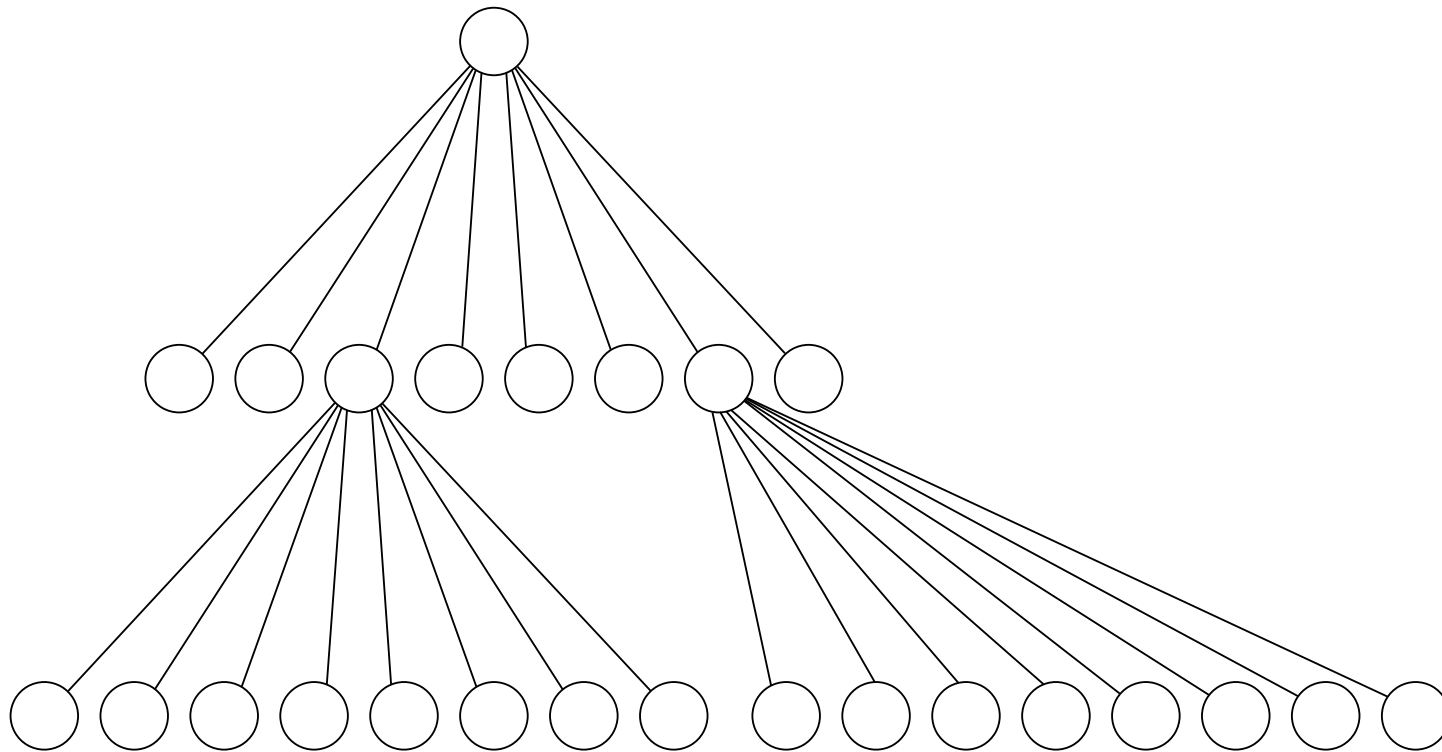
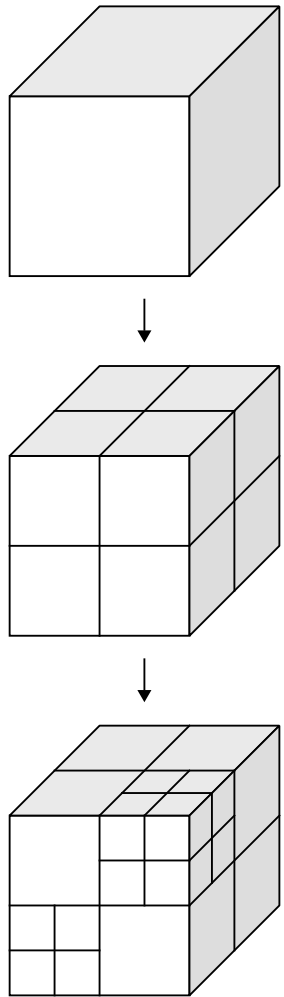
- Occlusion culling : éliminer du traitement tout ce qui n'est pas visible
  - Hors champ
  - Caché par des murs/montagnes
    - Utilisé pour l'affichage
- Détection de collision
- Ray tracing/casting :
  - Picking/sélection au clic de souris



# Construction habituelle

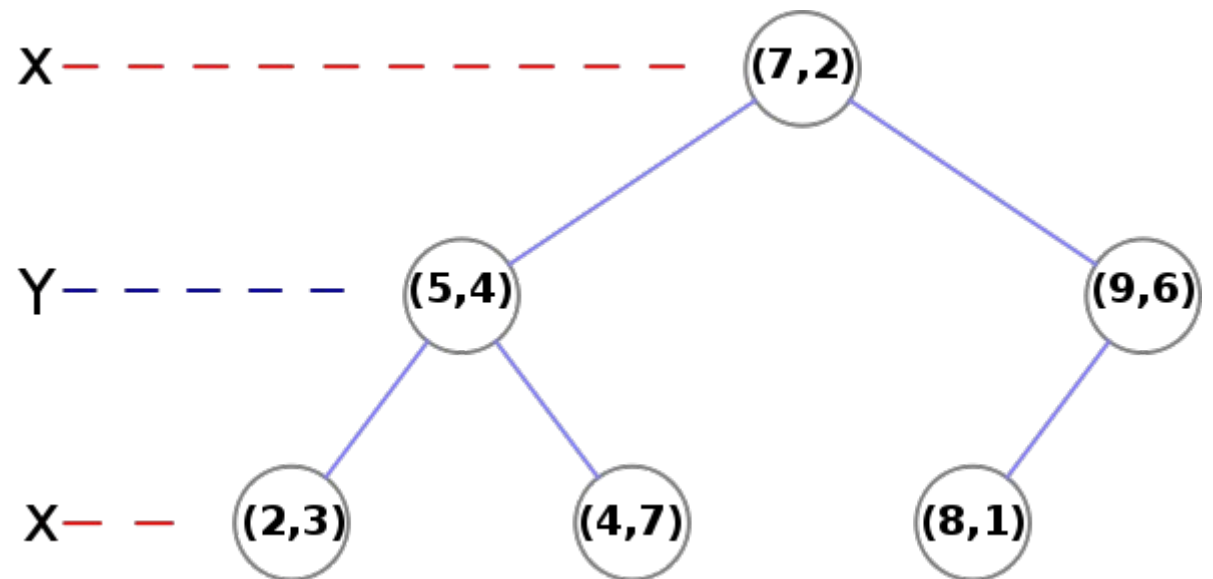
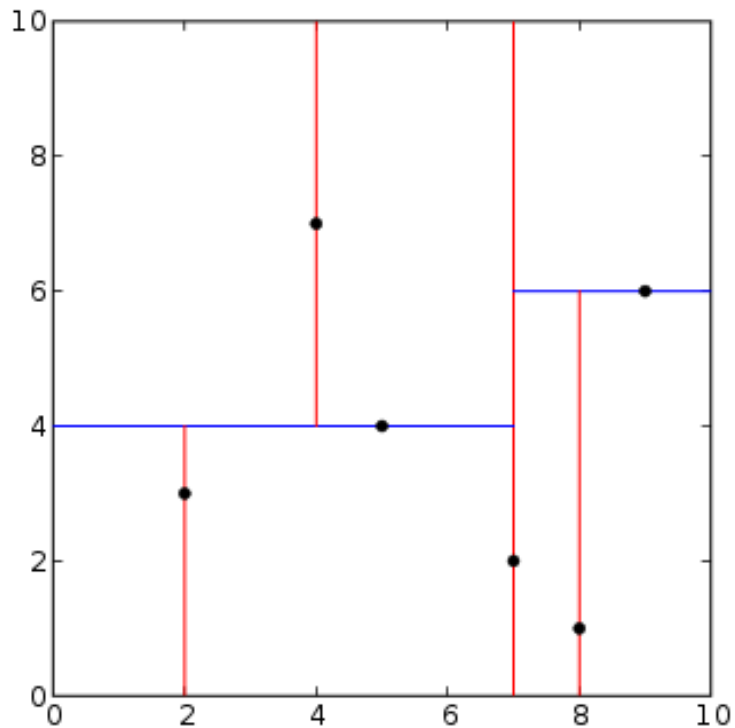
- Aux objets sont associés une représentation simplifiée
  - Généralement la boîte ou la sphère englobante
    - Le test d'intersection est peu coûteux (par rapport à tester les 30000 triangles du mesh)
  - Ces substituts doivent être conservatifs
    - Sinon artefacts visuels
- Ces versions simplifiées sont les briques élémentaires manipulées par la structure d'accélération

# Octree

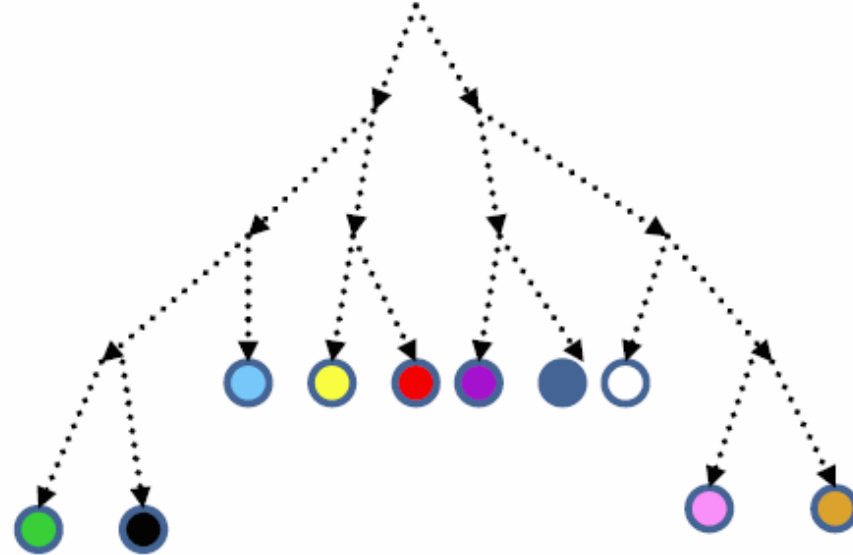
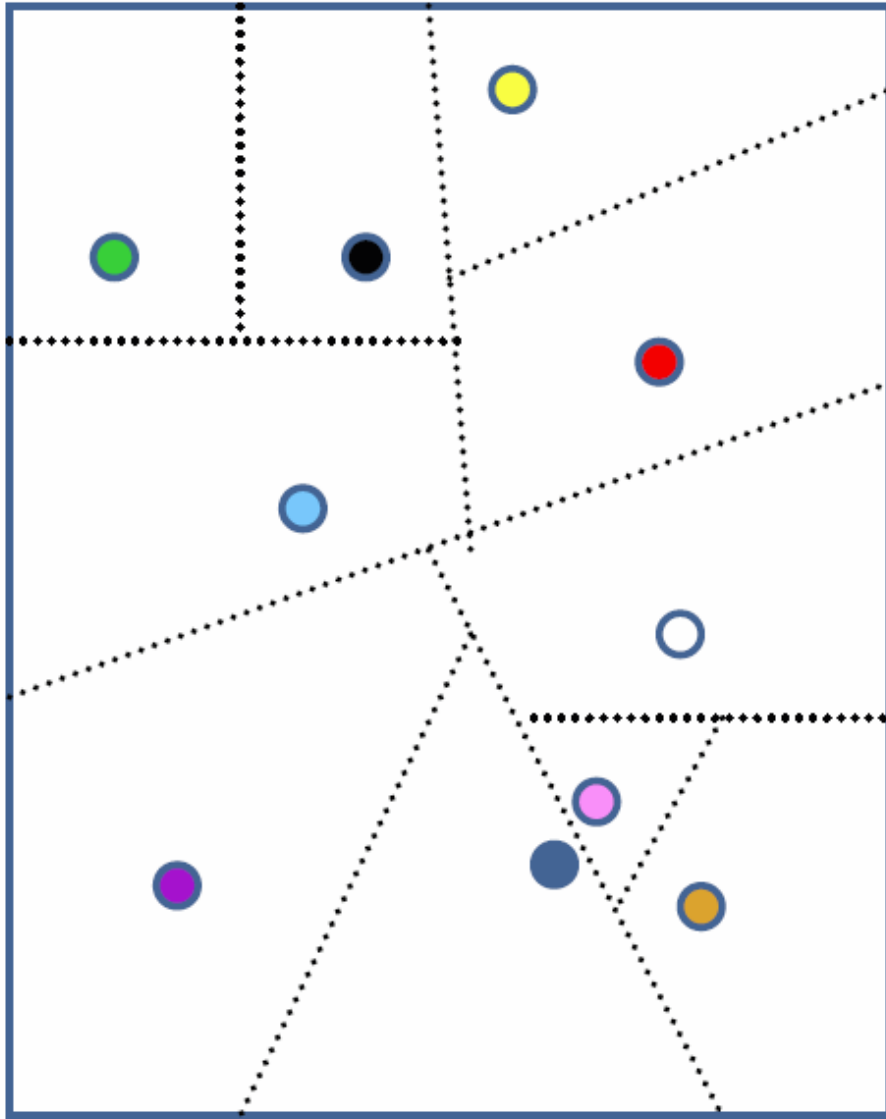


(source : wikipedia)

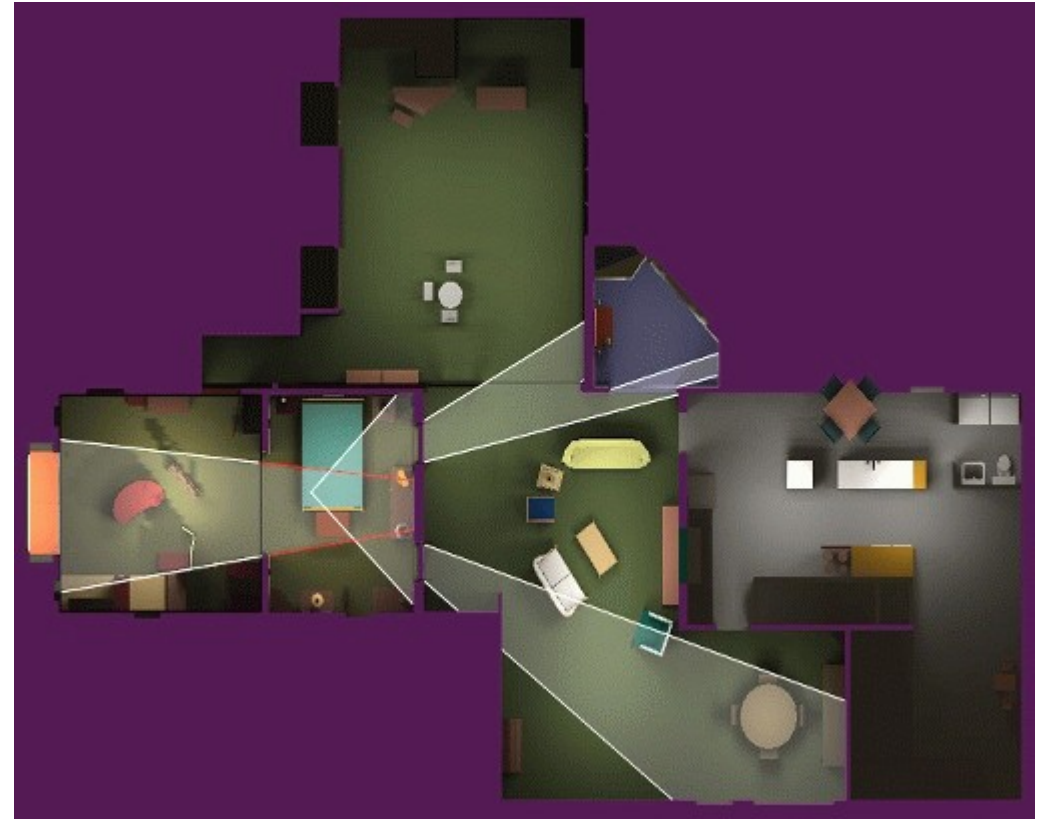
# k-d tree



# BSP tree



# Portal / PVS

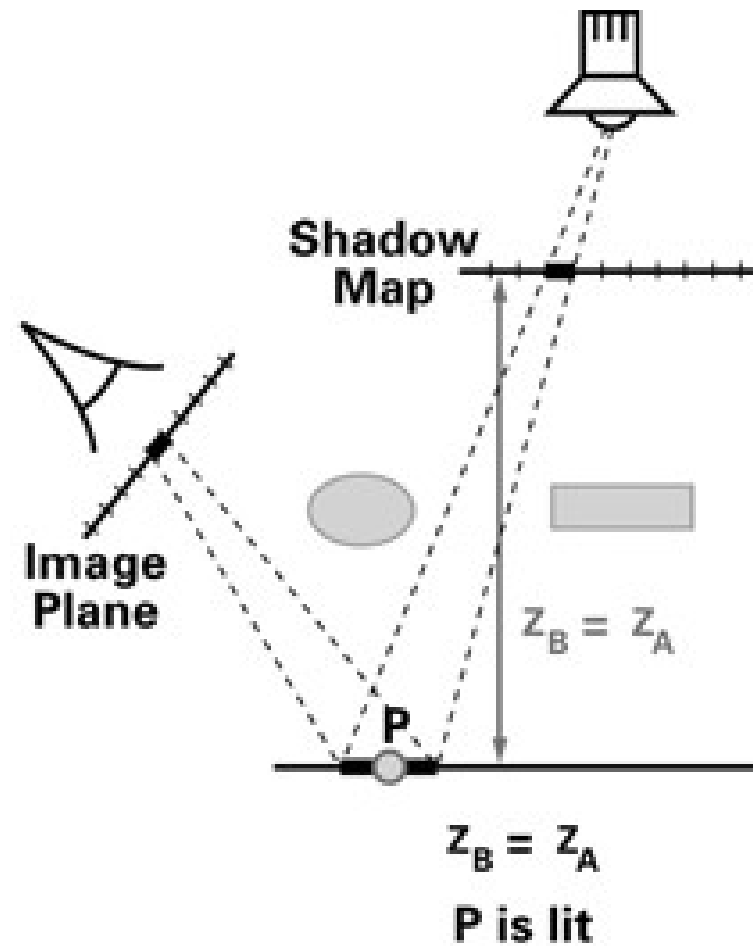
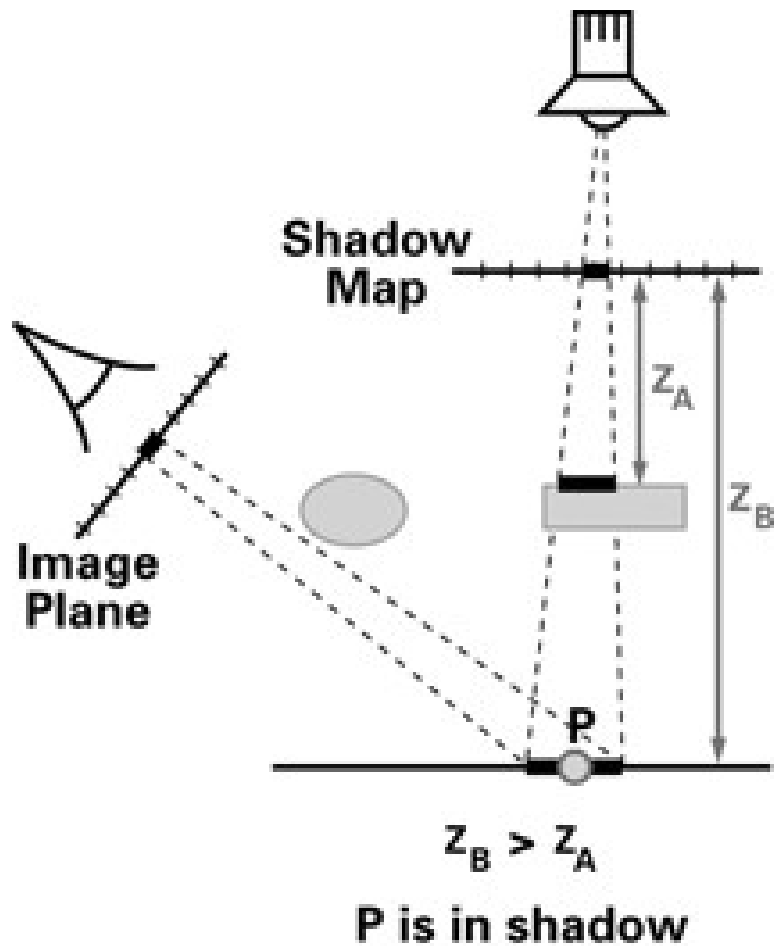


# Shadow mapping

# Problématique : les ombres

- Ombre = absence de lumière
  - Occlusion par des objets
- Comment savoir si l'on est dans l'ombre ?
  - Force brute : vérifier tous les objets pour voir s'il y en a qui bloque la lumière → impossible en temps réel
  - Moyen plus économique : shadow mapping
    - Il existe aussi la technique des shadow volumes

# Explications





# Pourquoi le shadow mapping ?

- Avantages :
  - Mêmes principes que la rasterisation de la scène pour l'affichage, mais vu de la lumière
  - Simplicité de l'algorithme (dans une certaine mesure)
- (Quelques) Inconvénients :
  - Crénelage
  - Imprécisions
  - Coût en mémoire et en performance :
    - Un rendu complet et un buffer par source lumineuse
  - Voir :
    - <https://docs.microsoft.com/fr-fr/windows/win32/dxtecharts/common-techniques-to-improve-shadow-depth-maps>
    - <https://docs.microsoft.com/fr-fr/windows/win32/dxtecharts/cascaded-shadow-maps>
- Y'a pas mieux pour le moment

# Animation

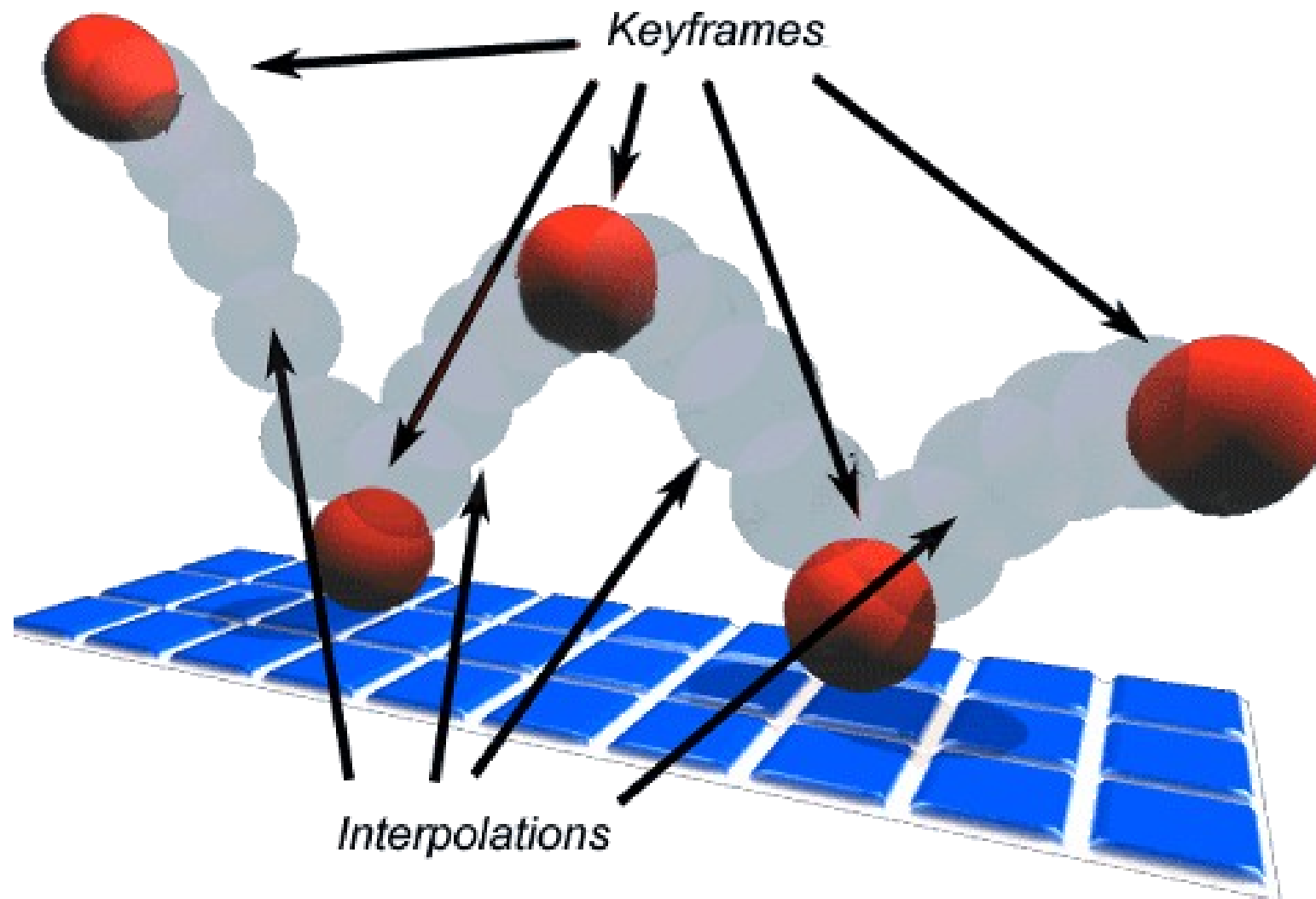
# Animer, c'est quoi ?

- Faire évoluer dans le temps des valeurs
- Quelques valeurs à animer :
  - Position, rotation
    - mouvement des objets
  - Couleurs, positionnement des textures
    - change l'aspect des objets
  - Positions des sommets
    - déformation des objets
- Morphing
- Skeletal animation / Skinning
  - Avec l'aide d'un squelette/d'une armature support

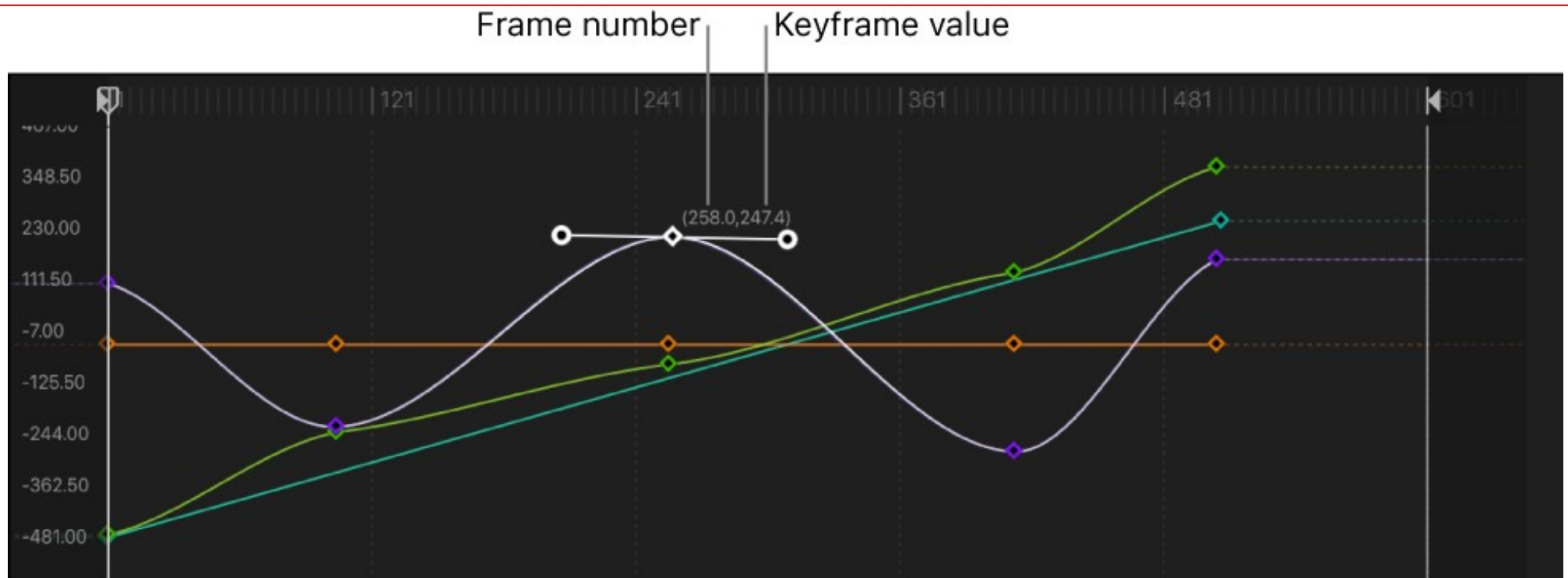
# Comment animer ?

- Animation procédurale
  - À partir de formules mathématiques, courbes, fonctions d'évolution dans le temps :  $x = f(t)$
- Animation par keyframe
  - Valeurs prédéfinies à des instants  $t$ , avec une interpolation entre ces instants
- Motion capture
  - Valeurs extraites à partir de capture du monde réelle
- Cinématique inverse
  - Valeurs calculées à partir de contraintes mécaniques (on s'approche du moteur de physique)

# keyframing

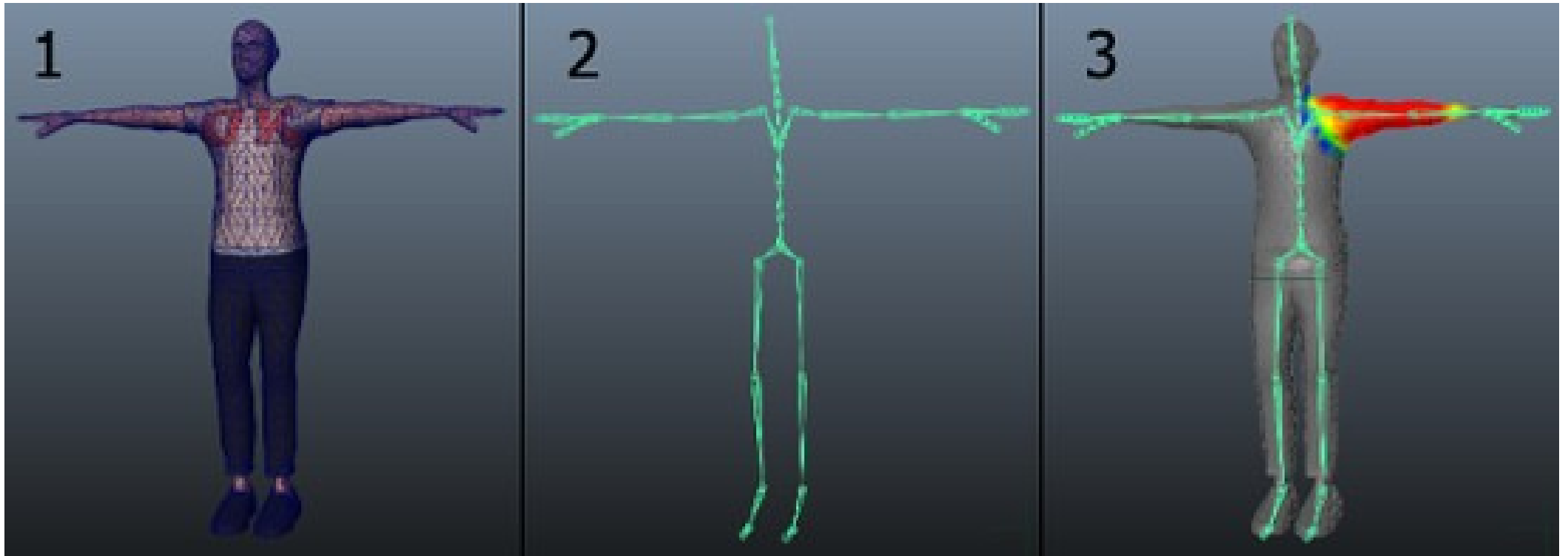


# keyframing



Animated				
<input checked="" type="checkbox"/>	▼ Rectangle			
<input checked="" type="checkbox"/>	■ Transform.Position.X	589.39	⏮	◆
<input checked="" type="checkbox"/>	■ Transform.Position.Y	113.58	↔	◆ ▼
<input checked="" type="checkbox"/>	■ Transform.Position.Z	0	⏮	◆
<input checked="" type="checkbox"/>	■ Blending.Opacity	77.33	⏮	◆

# Skinning/rigging



# Moteur de particules



# Pourquoi ?

- Cas spécifiques difficiles à reproduire de manière directe
  - Particulaires : gouttes de pluies, feux d'artifices, étincelles
  - « Brumeux » : fumées, nuages
  - Filaires : herbes, poils
- « Simulation » basique
  - Simplification physique, modèle approximatif
- Habillage travaillé : textures, couleurs, randomness
  - Puff (nuage, fumée)
  - Flare, scintillements, bloom (sprites, billboards)

# Moteur de physique

# Toujours reproduire le monde réel

- Simuler les règles de la mécanique pour produire des animations réalistes

# Quoi simuler ?

- Collisions, *collision detection*
- Mécanique du solide indéformable, *rigid body*
- Solides déformables, *soft body*
- Mécanique des fluides, *fluid*
- Fracturations
- Tissus
- Poils, cheveux

# Examples

- <http://brm.io/matter-js/demo/#mixed>
- <http://chandlerprall.github.io/Physijs/>
- <http://madebyevan.com/webgl-water/>

# Récapitulatif du module

# Les grandes notions

- Les APIs et les middleware 3D
- Le pipeline du GPU
  - Rastérisation
  - Shaders
- La géométrie des objets : maillages
- L'aspect des objets : réflectance des surfaces

# Pour aller plus loin

- Le module est une introduction succincte
  - Explorez par vous-même si ça vous intéresse
    - Étudiez, programmez !
    - Ce sont des connaissances que vous n'aurez pas l'occasion de développer dans un cursus d'enseignement standard (temps trop limité)
  - Ressources
    - Internet :
      - » <http://www.realtimerendering.com/>
    - Livres



# Compléments

# Modeleurs 3D

- Logiciels de création de données 3D
  - 3ds Max, Maya, Mudbox (Autodesk)
  - Sketchup
  - Rhino3D
  - ZBrush
  - Blender
  - ...

# Offline (non temps-réel)

- Applications médias passifs
  - cinéma, TV, animations, illustrations
- Domaine très différent du temps réel
  - Temps de calcul théoriquement illimité
  - Plus de possibilités d'effets
    - Global illumination
    - Caustics
    - Subsurface scattering
    - Participating media (effets volumétriques)
    - Fluides (Navier-Stokes)
- Algorithmes différents
  - Radiosité
  - Ray tracing
    - (bidirectional) path tracing, Photon mapping, MLT (Metropolis light transport), etc.