Tahsin Rossanaly - IR3

Plan de la présentation

- Présentation de XNA
- Architecture et fonctionnement de XNA
- Éléments d'architecture d'un jeu
- Interaction avec l'environnement
- Gestion de la 3D
- Informations pratiques

Présentation de XNA

- XNA = XNA 's Not Acronymed :-)
- Ensemble d'outils facilitant le développement de jeux vidéo pour :



Présentation de XNA

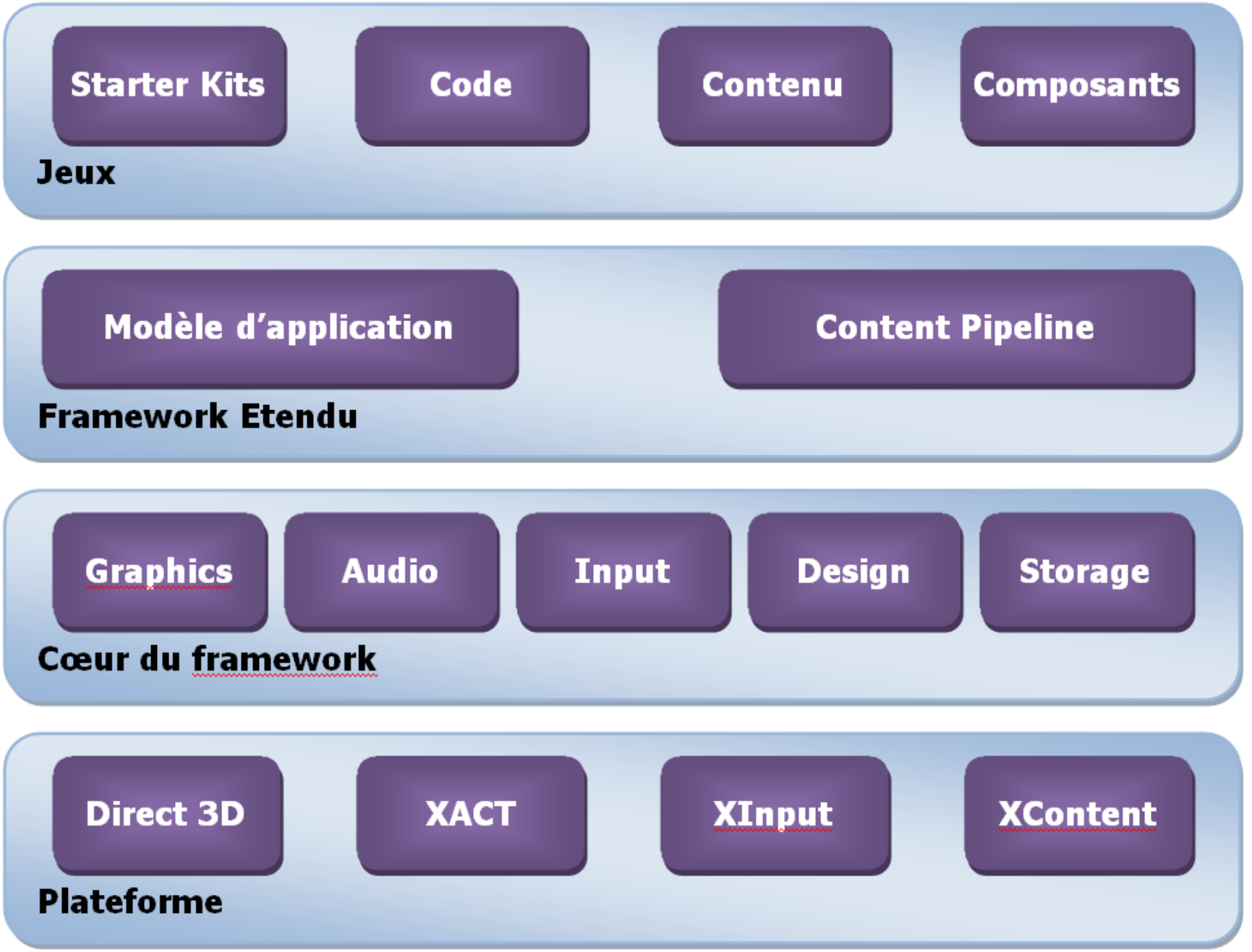
- XNA fournit :
  - Un framework .net
  - Des outils d'intégration de contenu
  - De la documentation
- XNA nécessite Visual C#

Présentation de XNA

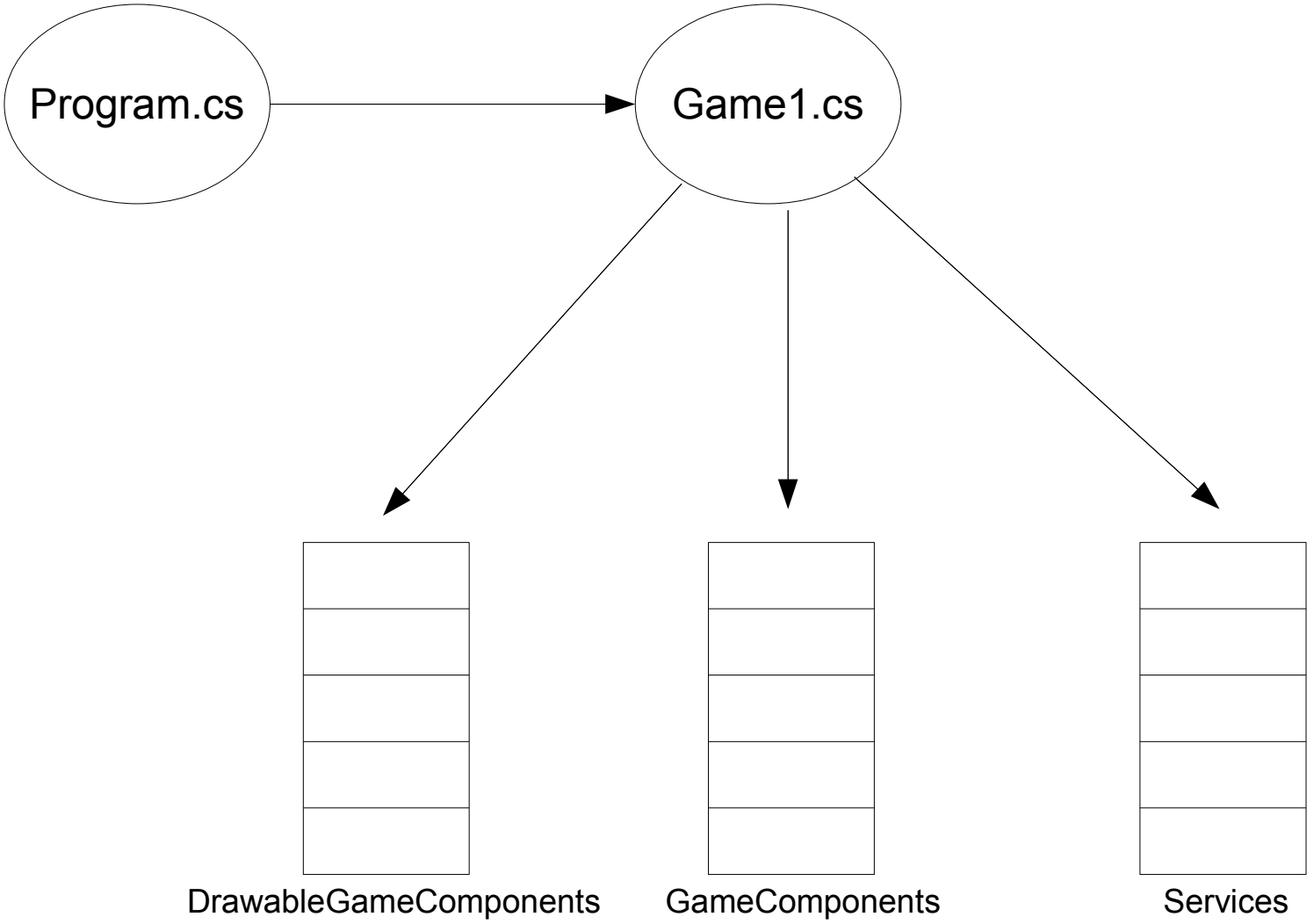
- Les avantages de XNA
  - Multi-plateforme (au sens Microsoft)
  - Facilite le développement des jeux vidéo
  - Accessibilité
  - Visibilité

- Les inconvénients de XNA
  - Uniquement en C#
  - Que du Microsoft!
  - Actuellement, pas de Kinect.
  - Visibilité

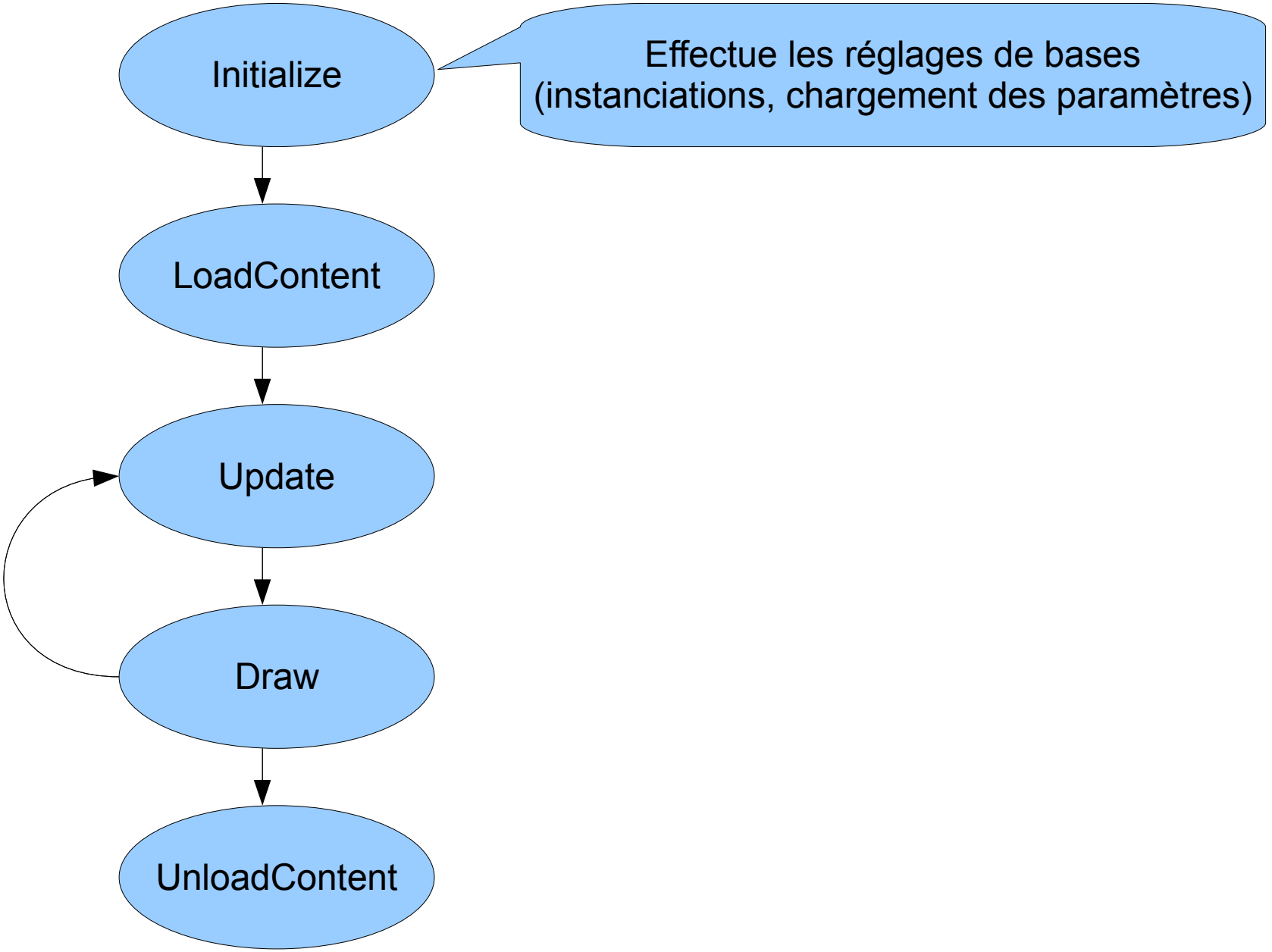
Architecture et fonctionnement de XNA



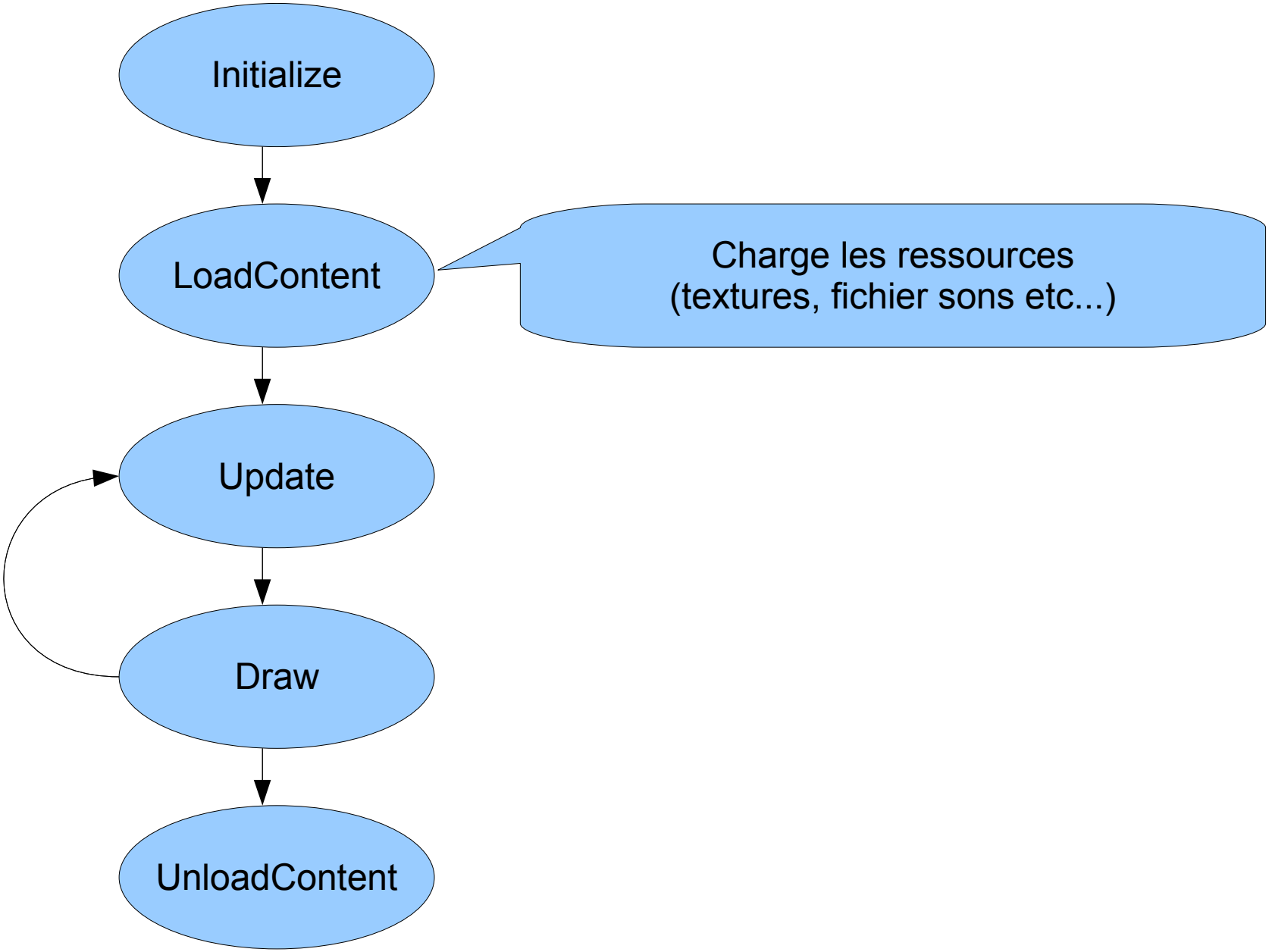
Architecture et fonctionnement de XNA



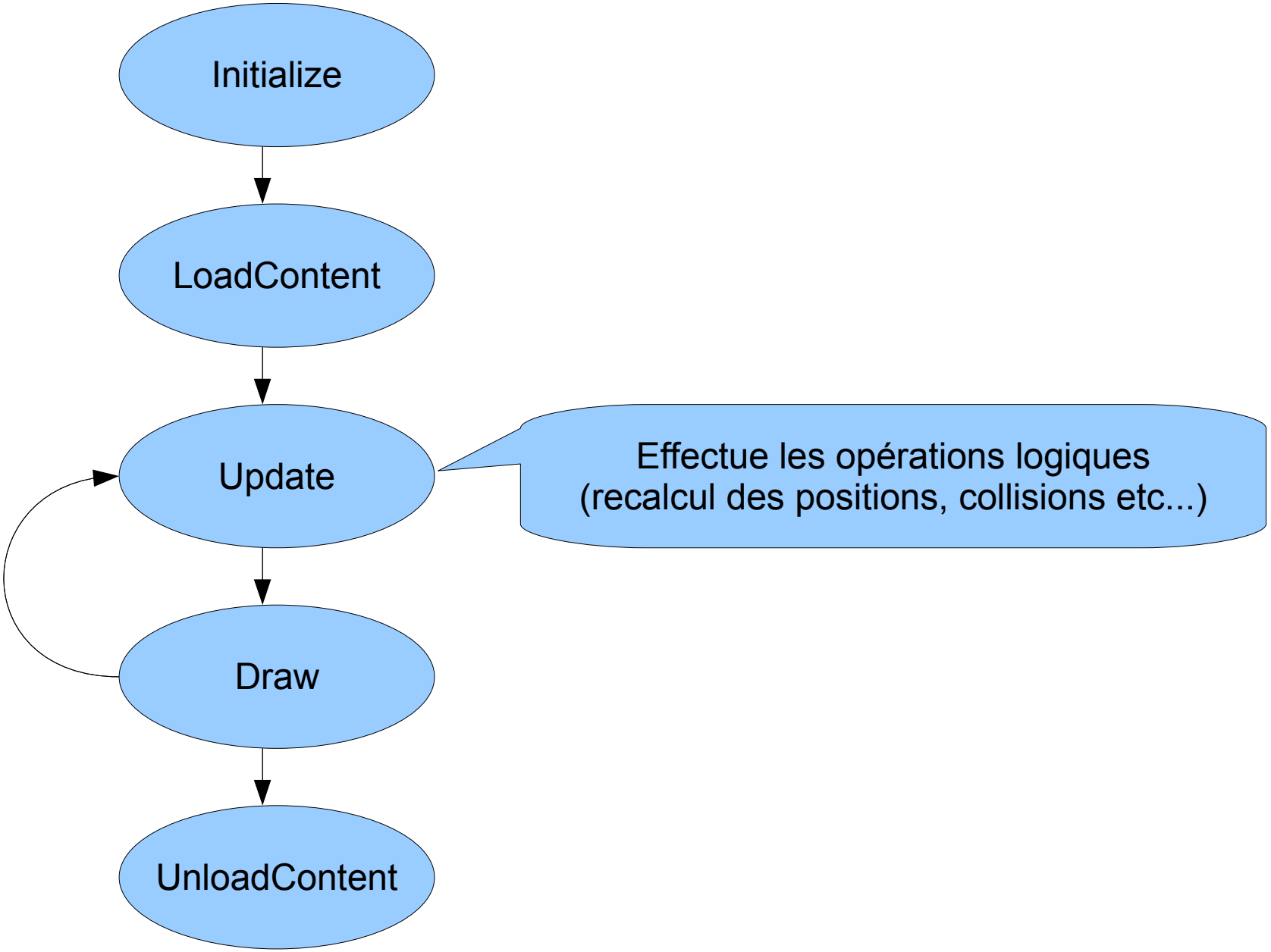
Architecture et fonctionnement de XNA



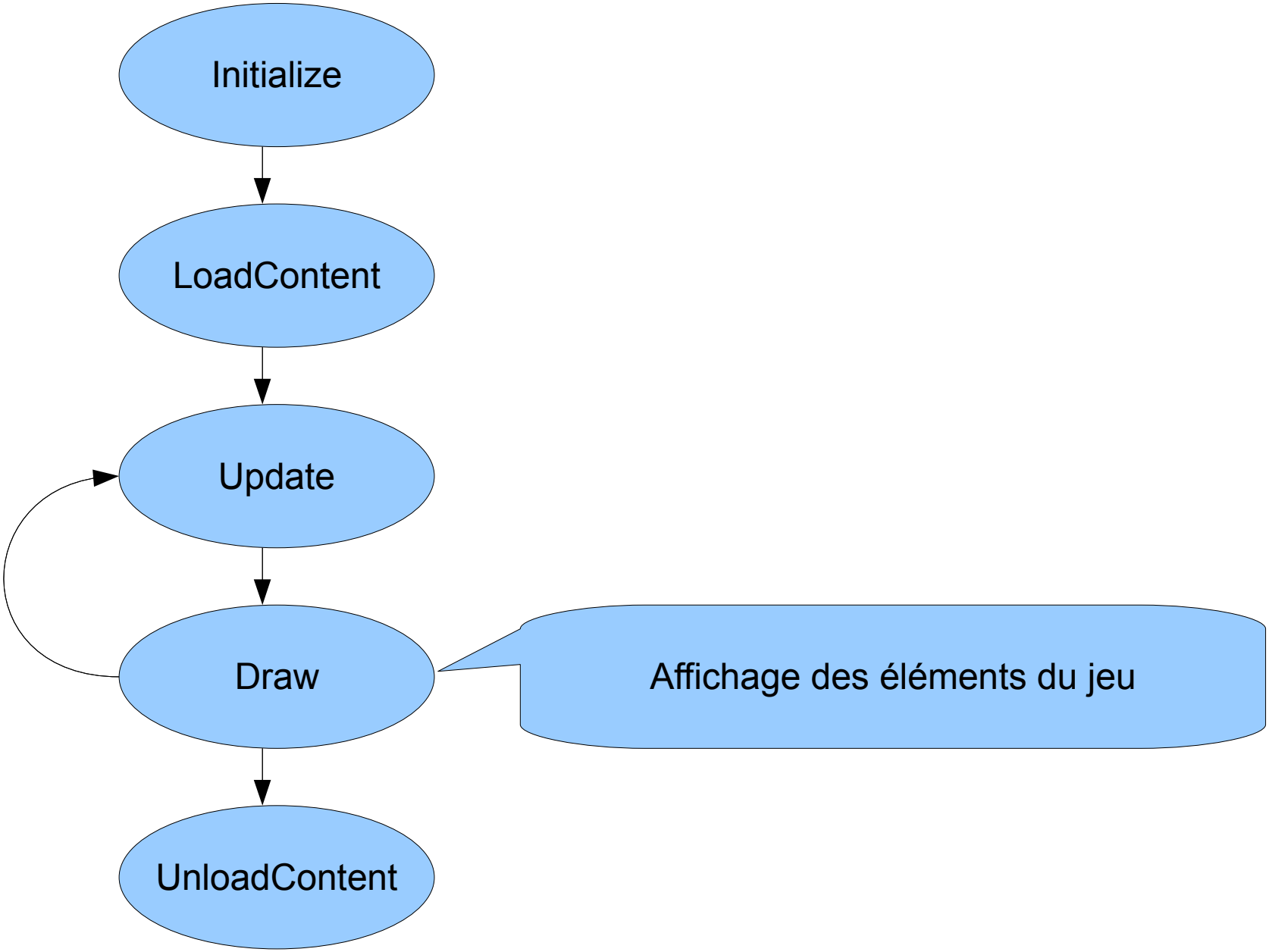
Architecture et fonctionnement de XNA



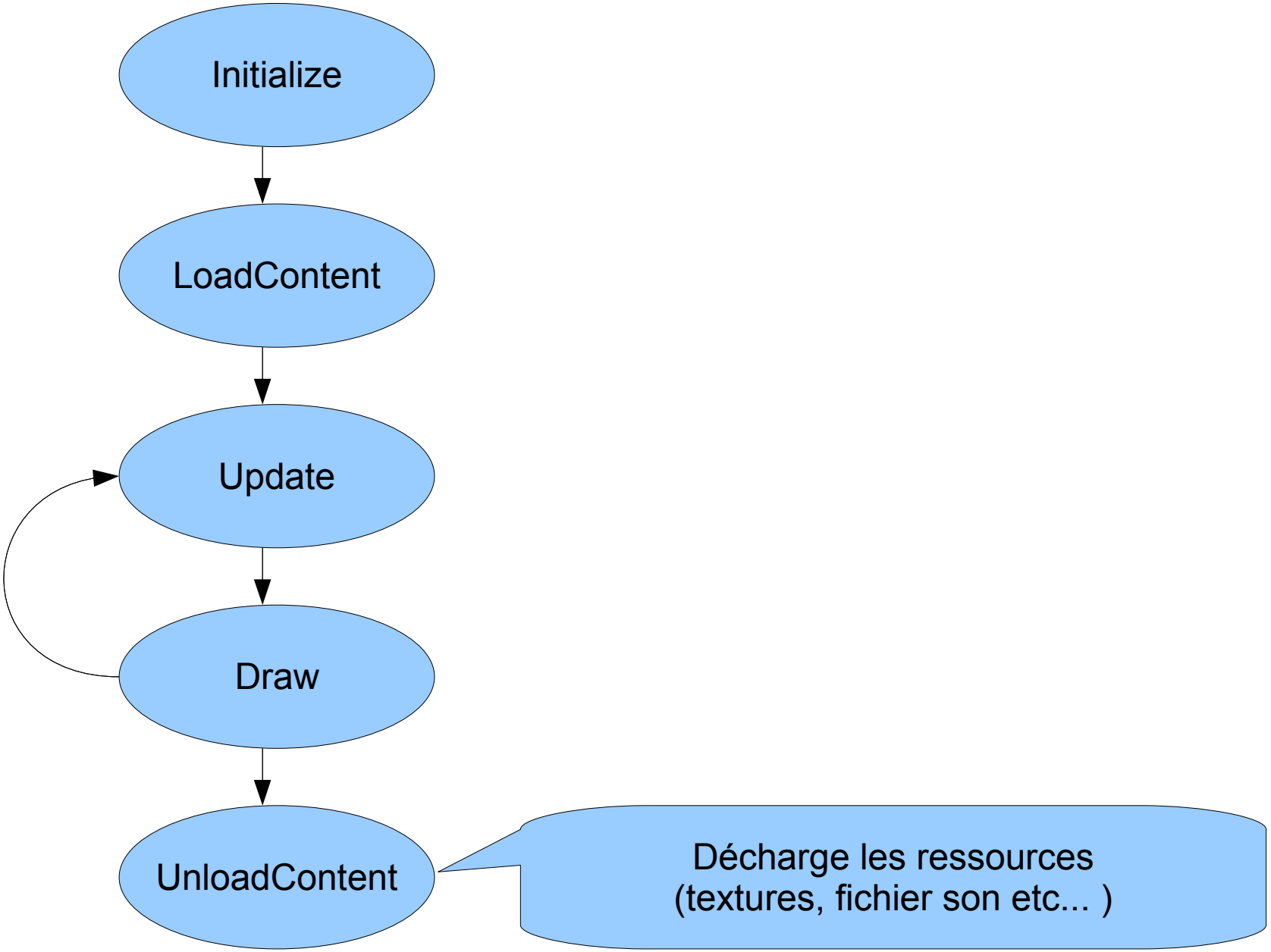
Architecture et fonctionnement de XNA



Architecture et fonctionnement de XNA



Architecture et fonctionnement de XNA



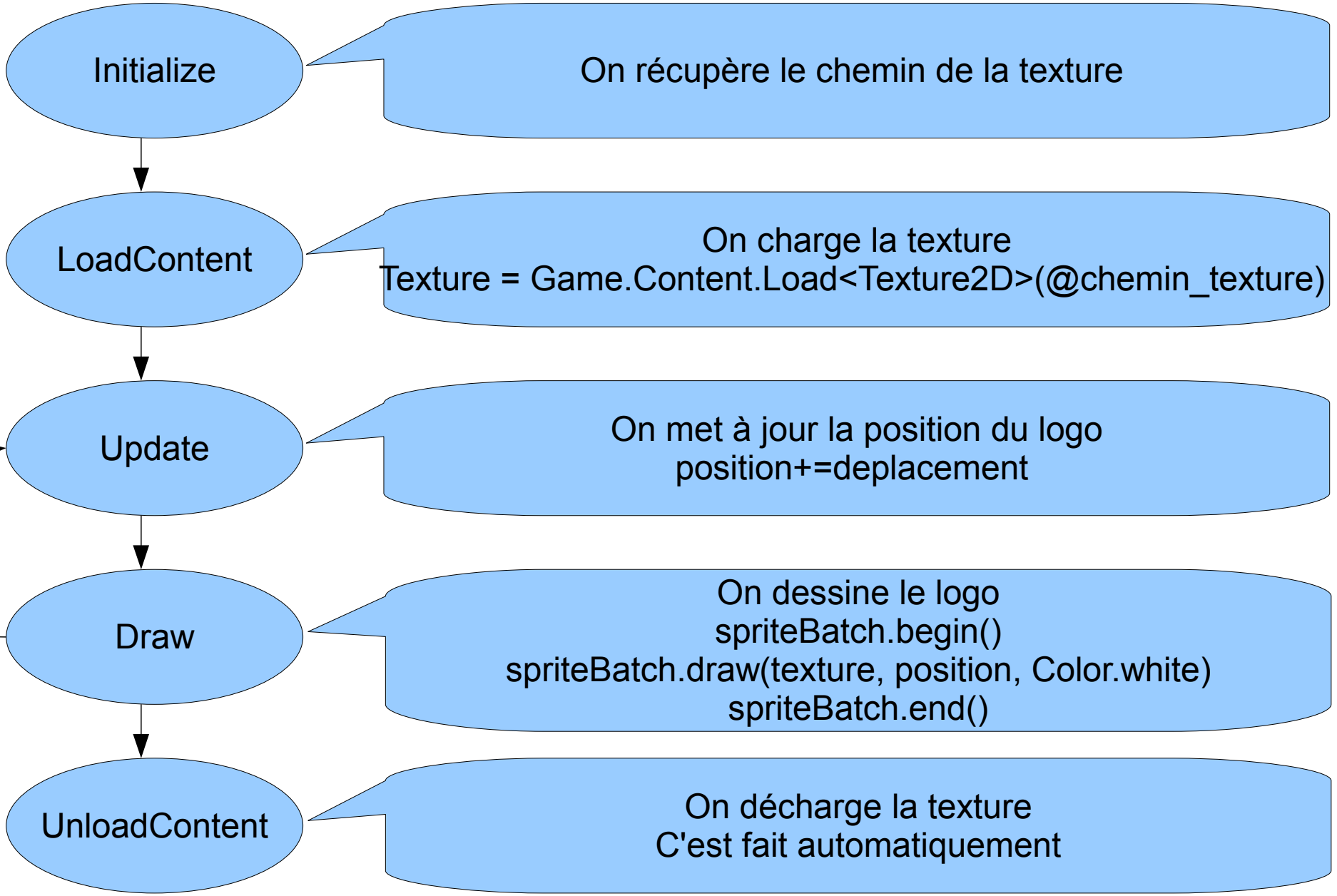
Architecture et fonctionnement de XNA

- Structure commune aux différents composants
- Automatisation des traitements des composants
  - Instancier le composant dans Game1 suffit

Architecture et fonctionnement de XNA

- Exemple : affichage du logo de XNA qui se déplace.
- Création d'une classe qui dérive de DrawableGameComponent
- On l'instancie au niveau de Game1, dans la méthode initialize()
  - DGC logo = new logo_xna(10,10,this);
- C'est le logo qui s'ajoute lui même.

Architecture et fonctionnement de XNA



Architecture et fonctionnement de XNA

- Garder à l'esprit la diversité des machines
  - Utilisation d'un gametime pour un fonctionnement uniforme
- Diverses opérations possibles
  - N'afficher qu'une partie de la texture
  - Redimensionner la texture
  - Ajouter une teinte
  - Afficher du texte
  - ...

Architecture et fonctionnement de XNA

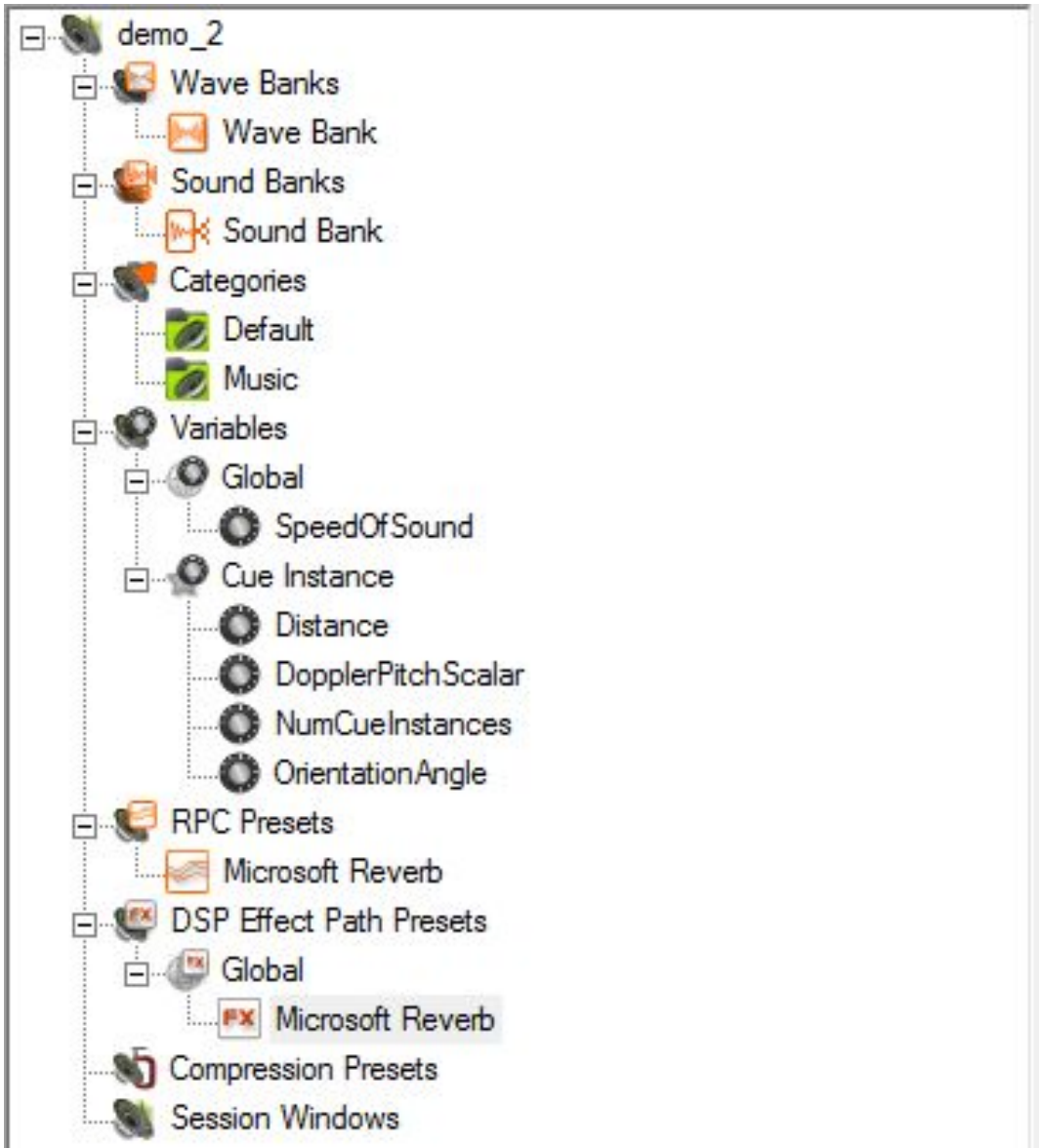
- Pour gérer le son, il existe deux méthodes :
  - Utiliser l'API SoundEffect (similaire au fonctionnement des images)
  - Utiliser l'API XACT

Architecture et fonctionnement de XNA

- XACT est un framework permettant de créer des projets audio.
- Possibilité de compresser, de modifier la vitesse du son, d'ajouter des effets de réverbération

Architecture et fonctionnement de XNA

- Un aperçu des possibilités



Architecture et fonctionnement de XNA

- Pour utiliser notre projet Audio :

```
engine = new AudioEngine(@"Content\demo_2.xgs");
soundBank = new SoundBank(engine, @"Content\Sound Bank.xsb");
waveBank = new WaveBank(engine, @"Content\Wave Bank.xwb");
```

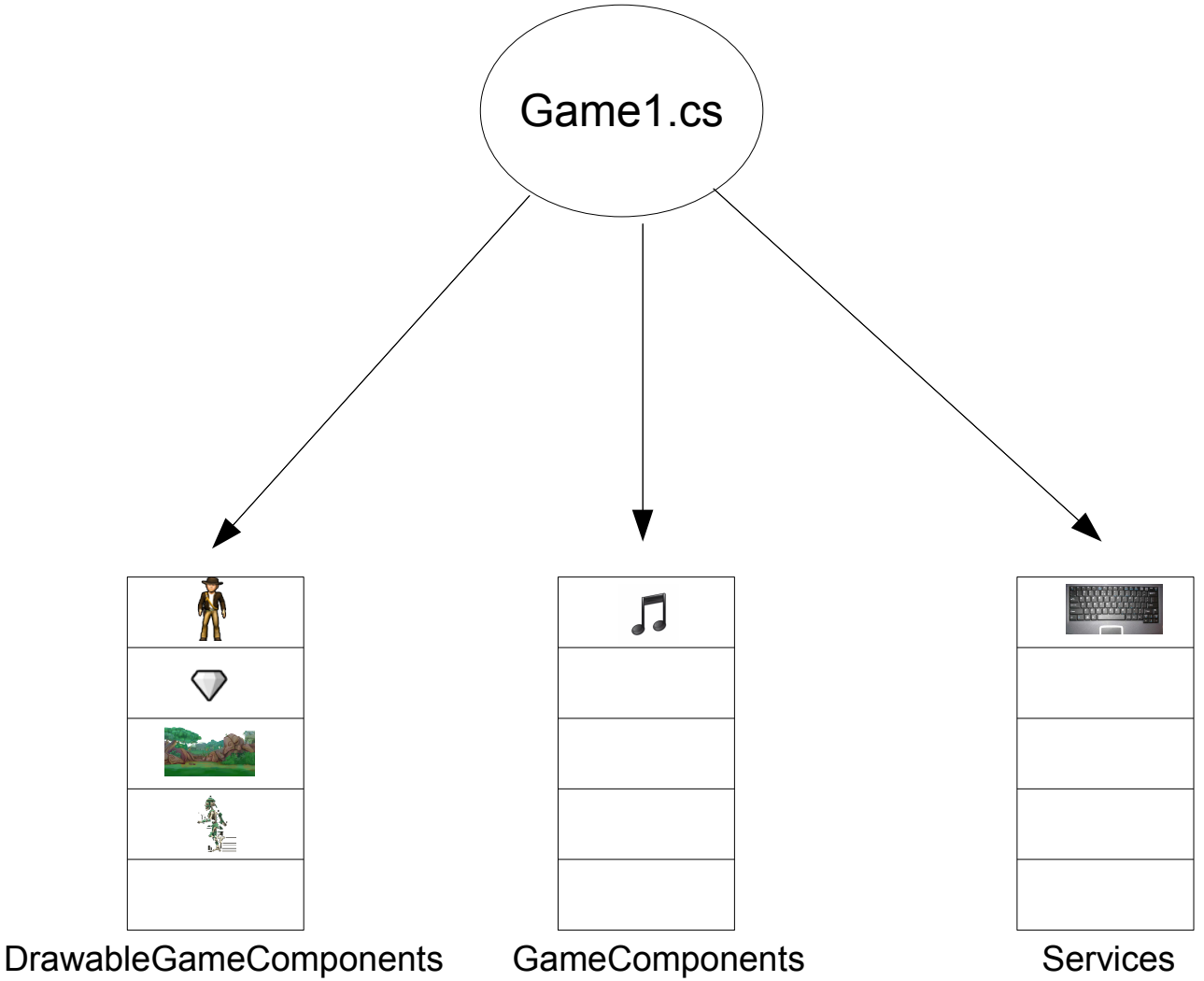
```
engine.Update();
soundBank.PlayCue("urg2");
```

Éléments d'architecture d'un jeu

- XNA fournit un jeu déjà complet : Platformer
- Aucune architecture imposée
  - Possibilité d'automatiser les différents traitements
  - Ou les gérer manuellement
- Une petite démonstration du jeu

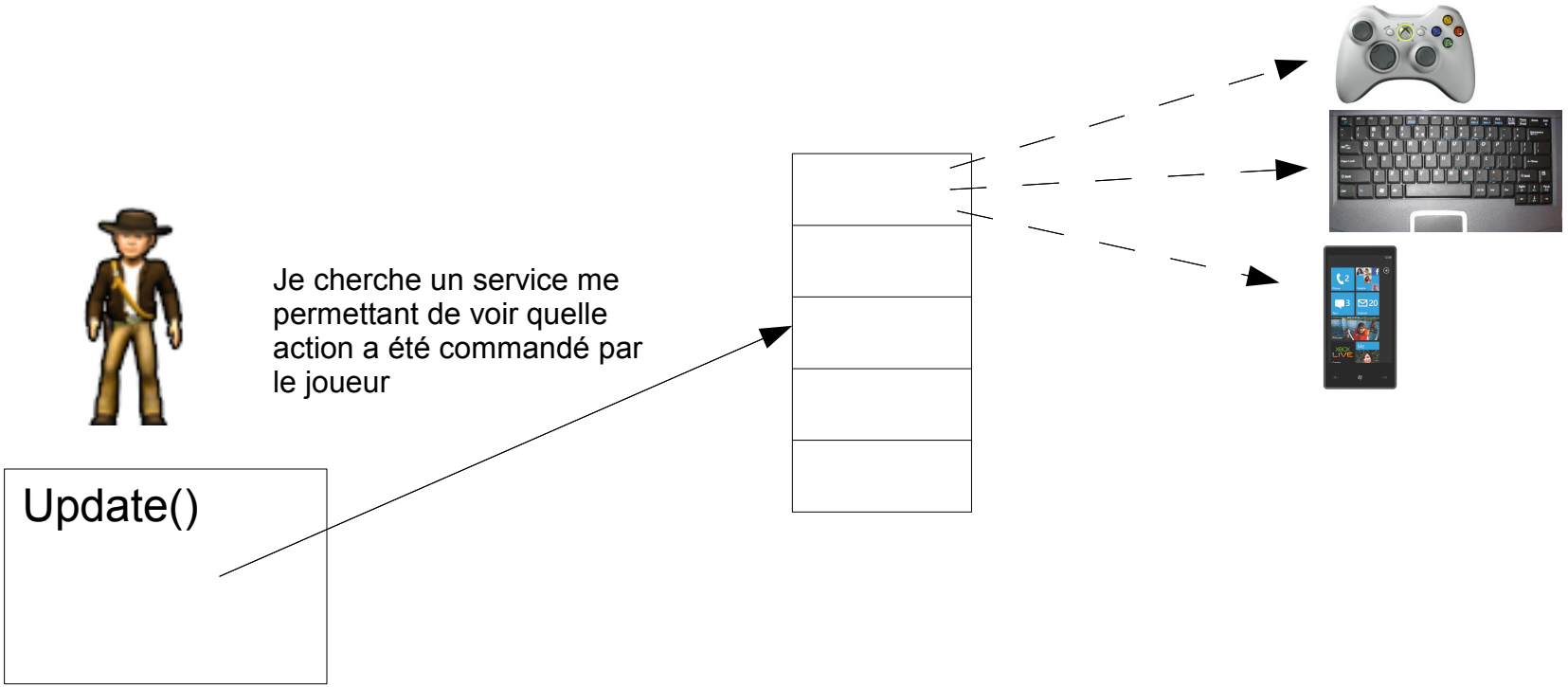
Éléments d'architecture d'un jeu

- Automatisation des traitements



Éléments d'architecture d'un jeu

- Utilisation des services pour le contrôle du jeu
- Nécessité de séparer les contrôles



Éléments d'architecture d'un jeu

- Avantages des services :
  - On n'utilise pas de méthodes statiques
  - Une seule instanciation
  - On ne passe plus les contrôles en paramètres

Eléments d'architecture d'un jeu

- XNA ne possède pas de méthodes pour gérer la physique (collisions, gravité etc...)
  - Il faut la développer soit même
  - Utiliser un des nombreux moteurs existants(JiggleX, BulletX, FarseerPhysics...)

Éléments d'architecture d'un jeu

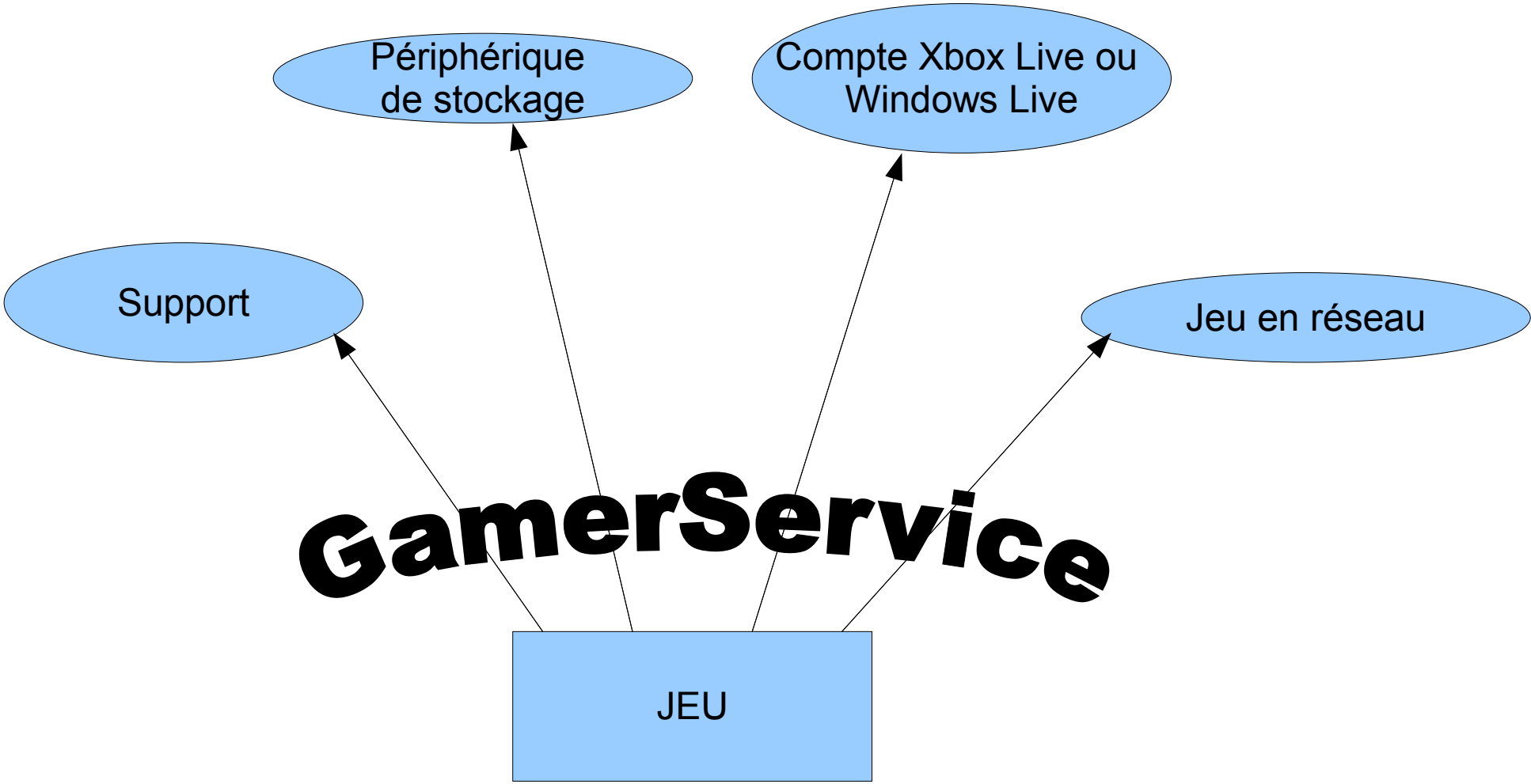
- Exemple d'utilisation du moteur physique FarseerPhysics
  - Création d'un PhysicsSimulator
  - On le passe aux éléments graphiques de notre jeu
  - Ceux-ci auront chacun un corps (body) qui leur permettra de les gérer suivant la gravité, les éléments qui appliquent une force dessus

Éléments d'architecture d'un jeu

- Exemple de code

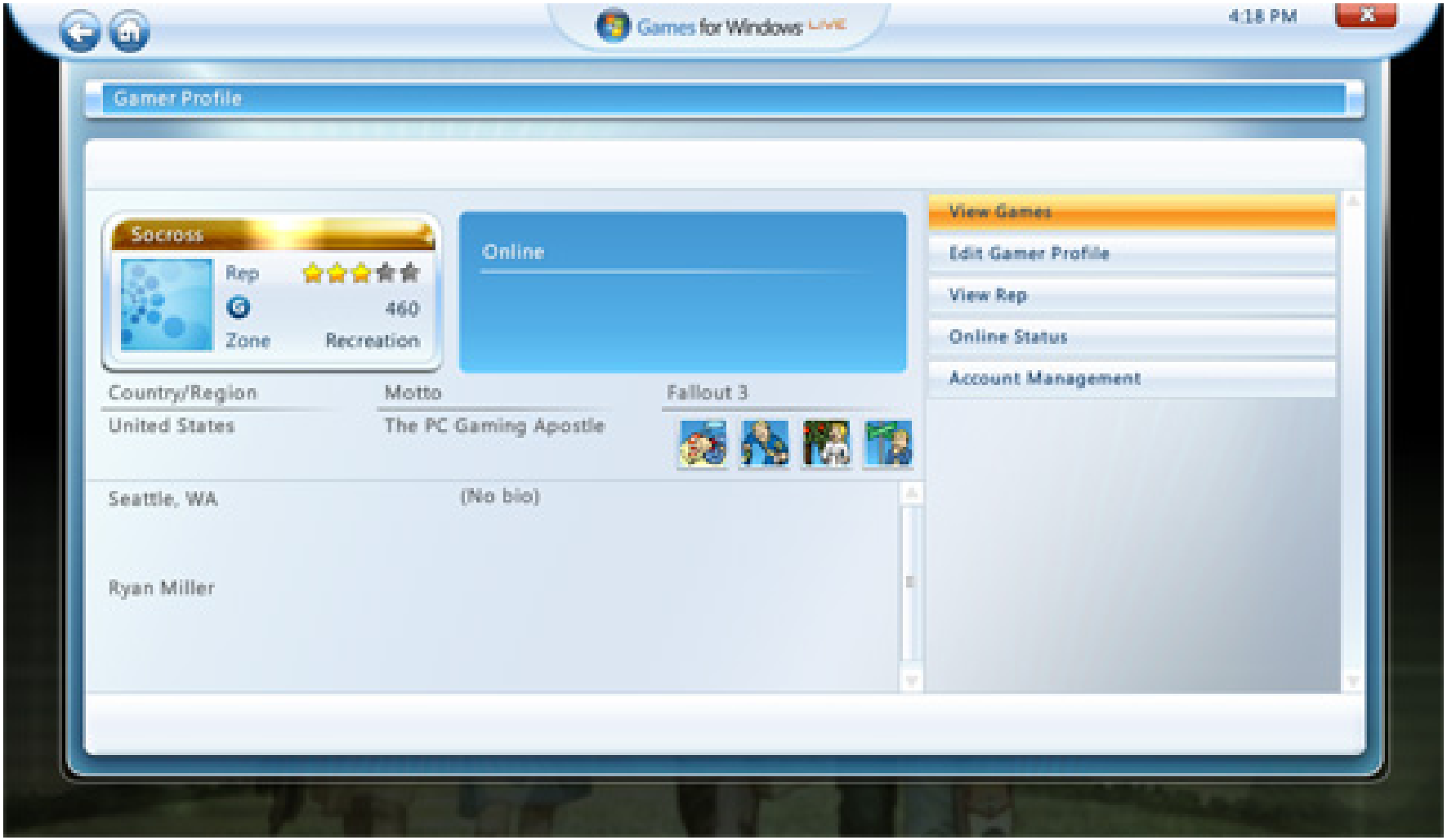
```
physicsSimulator = new PhysicsSimulator(new Vector2(0,750));
Hero = new Hero(physicsSimulator);


body = BodyFactory.Instance.CreateRectangleBody(physicsSimulator, 32, 32, 1);

body.Rotation += 0.05f;
body.ApplyForce(new Vector2(10,10));
```

Interaction avec l'environnement



Interaction avec l'environnement

- Afficher sa GamerCard



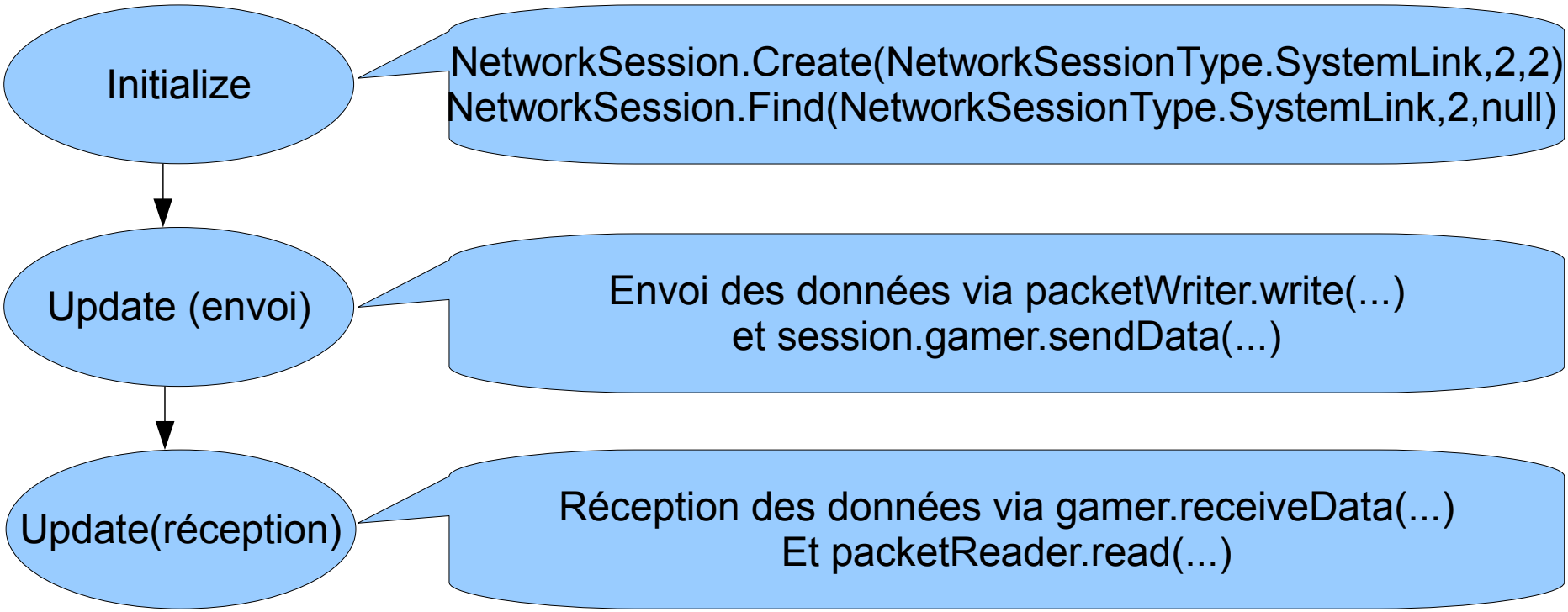
Interaction avec l'environnement

- Pour afficher la gamer Card :

```
this.Components.Add(new GamerServicesComponent(this));

Guide.ShowSignIn(1, false);

if (!Guide.IsVisible)
{
    try
    {
        Guide.ShowGamerCard(PlayerIndex.One, SignedInGamer.SignedInGamers[0]);
    }
    catch (Exception e)
    {
        Guide.BeginShowMessageBox(PlayerIndex.One, "Erreur", e.Message, new Str
    }
}
```

Interaction avec l'environnement

- Gestion du jeu en réseau (local ou en ligne)



Interaction avec l'environnement

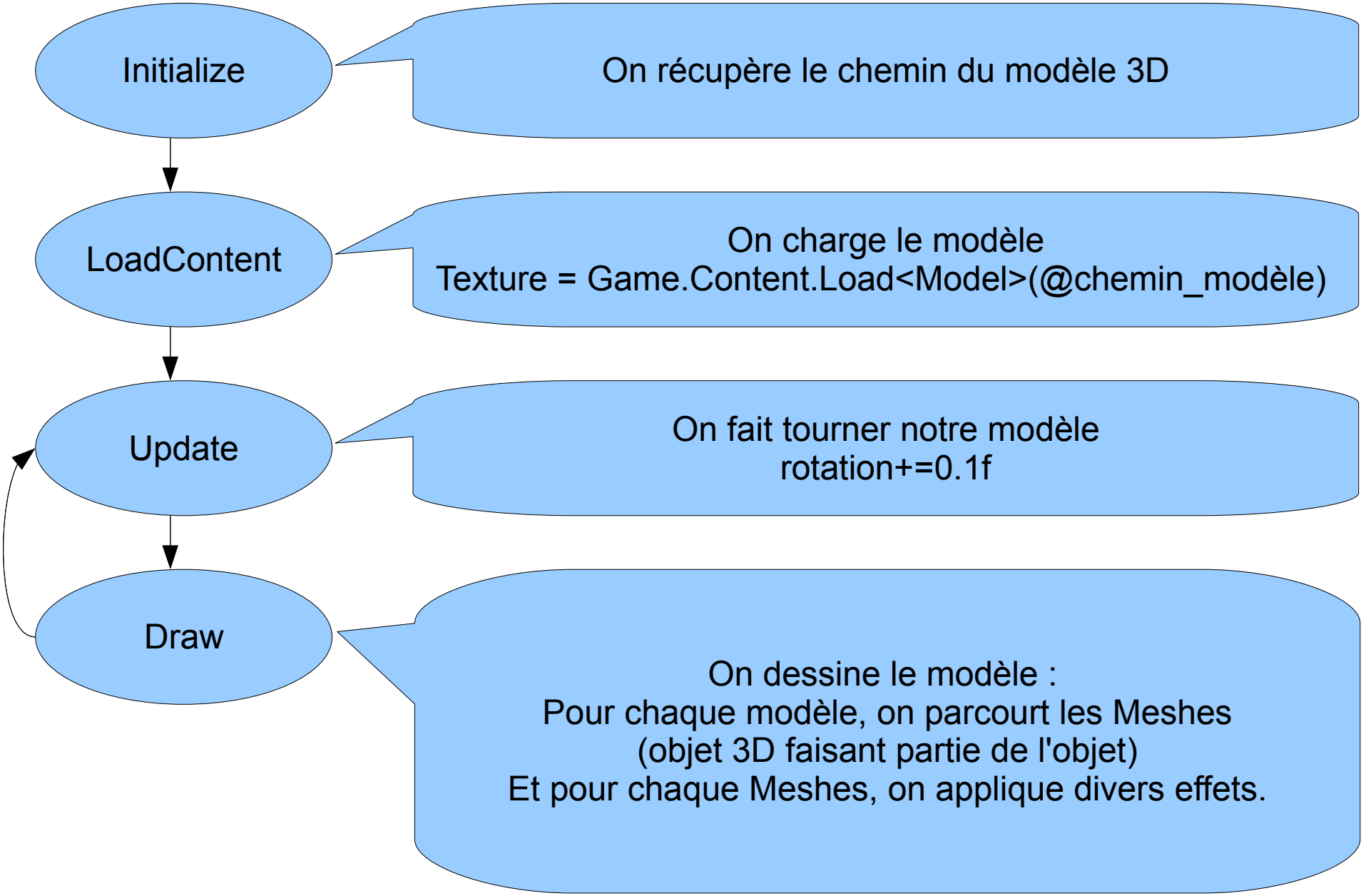
- Aucun besoin de synchronisation, ni de threads
- Choix de la politique d'envoi (fiable, non fiable, ordonné, désordonné...)

Gestion de la 3D

- Diverses possibilités :
  - Gestion de la caméra
  - Rotation des objets
  - Divers effet(brouillard, lumière...)
  - Personnaliser les effets grâce à HLSL (High Level Shader Language)

```
struct VS_OUTPUT
{
 float4 Pos : POSITION;
 float3 Pshade : TEXCOORD0;
};

VS_OUTPUT main (float4 vPosition : POSITION)
{
 VS_OUTPUT Out = (VS_OUTPUT) 0;

 // Transform position to clip space
 Out.Pos = mul (view_proj_matrix, vPosition);

 // Transform Pshade
 Out.Pshade = mul (texture_matrix0, vPosition);

 return Out;
}
```

Gestion de la 3D



Gestion de la 3D

- HLSL permet de programmer des shaders.
- Les shaders sont des instructions envoyées à la carte graphique pour gérer divers effets 2D et 3D (lumière, ombre, brouillard etc...)
- XNA fournit déjà des shaders basiques (que nous utilisons sans nous en rendre compte)

Informations pratiques

- Quelques astuces pour la conception du jeu :
  - Développer d'abord pour PC (plus simple et de toute façon on peut réadapter pour les autres plateformes)
  - Une architecture suffisamment souple pour pouvoir facilement adapter le jeu à chaque plateforme.
  - Une architecture suffisamment souple pour pour pouvoir le faire évoluer facilement (version Noel, version Halloween etc...)
  - Un nom bien pensé

Informations pratiques

- Test sur Xbox 360
  - Téléchargement de XNA Game Studio Connect à partir de la console
  - Génération d'une clé de connexion
  - Ajout de la console Dans XNA Game Studio Device Center
  - Compilation
  - Exécution

Informations pratiques

- Mise en ligne pour Xbox 360 et Windows phone
  - Nécessite un abonnement payant au XNA Creators Club
  - Soumission du jeu
  - Phase de Peer Review
  - Mise à disposition du jeu
  - Recueil du bénéfice (70% du prix du jeu)

Conclusion

- XNA permet de faciliter le développement de jeux vidéo pour les plateformes Microsoft
- XNA possède une architecture modulaire.
- Une stratégie différente de celles de ses concurrents

Des questions?

Merci de votre attention!