Game Asset Manager

Ferdinand Majerech <kiithsacmp@gmail.com>

Intro

- System to manage game assets (sounds, 3D models, VA...)
- Web-based frontend
- Cross-platform backend (Java)
- Using existing technologies to avoid reinventing the wheel

Ease of use

- Made for artists, translators, composers…
- **Not** for programmers
- Customizable, potentially usable by modders
- No need to understand VCS or issue tracking
- These can be handled in background

Workflow

- Asset owner (e.g art director):
  - Specifies assets, dependencies, milestones
  - Assigns asset creators to assets
  - Reviews assets
- Asset creator (e.g. graphics artist):
  - Submits asset files
- Any user can download files, view if possible

VCS integration

- Bridging the gap between artists and programmers
- DVCS-style workflow with one "official" repo but
- Local changes possible
- Git backend
- Extensions possible (Other VCSs, databases)

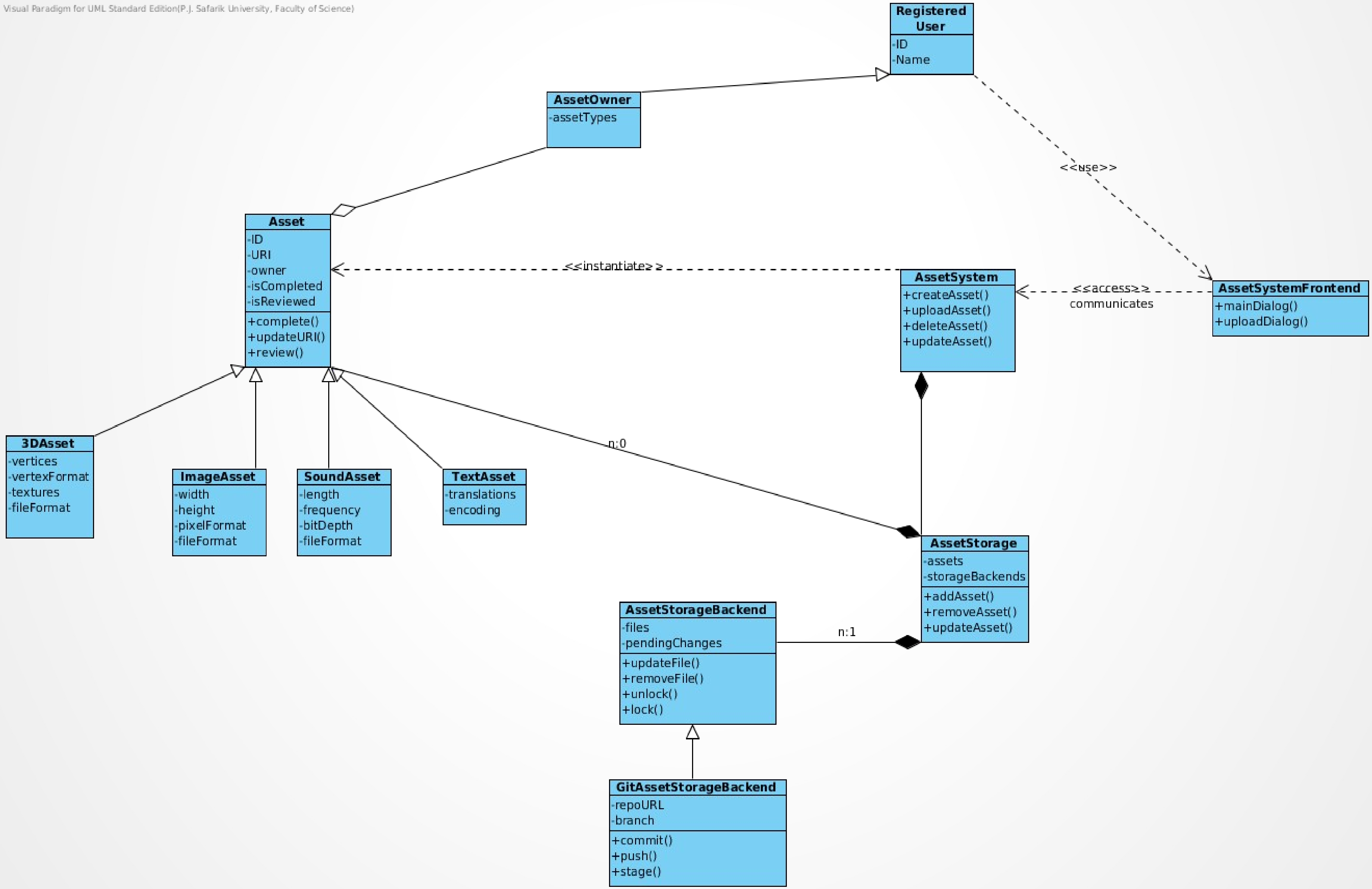
Asset dependency management

- Assets might depend on other assets (e.g. 3D model - textures)
- Dependencies must be met before an asset is complete.
- Milestones (finishing a campaign, DLC, etc.)

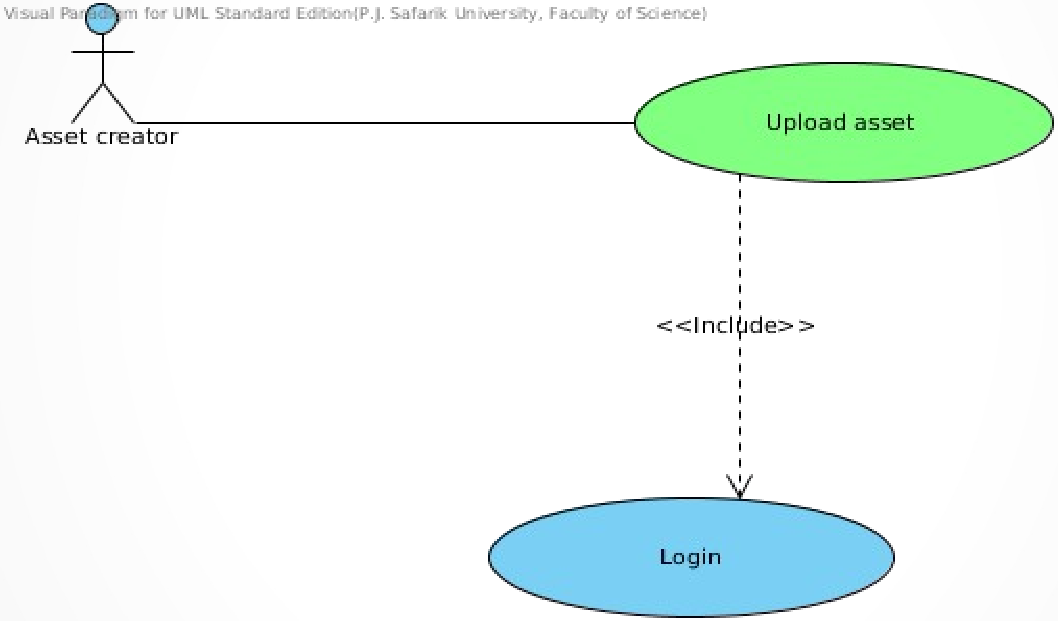
Game asset packaging

- Packaging assets into game data archives
- Generating VFS hierarchy
- Game(mod) archives from milestones
- Patch(delta) archives
- Preprocessing for various platforms
  - Textures: S3TC/PVRTC
- Package review (release manager).

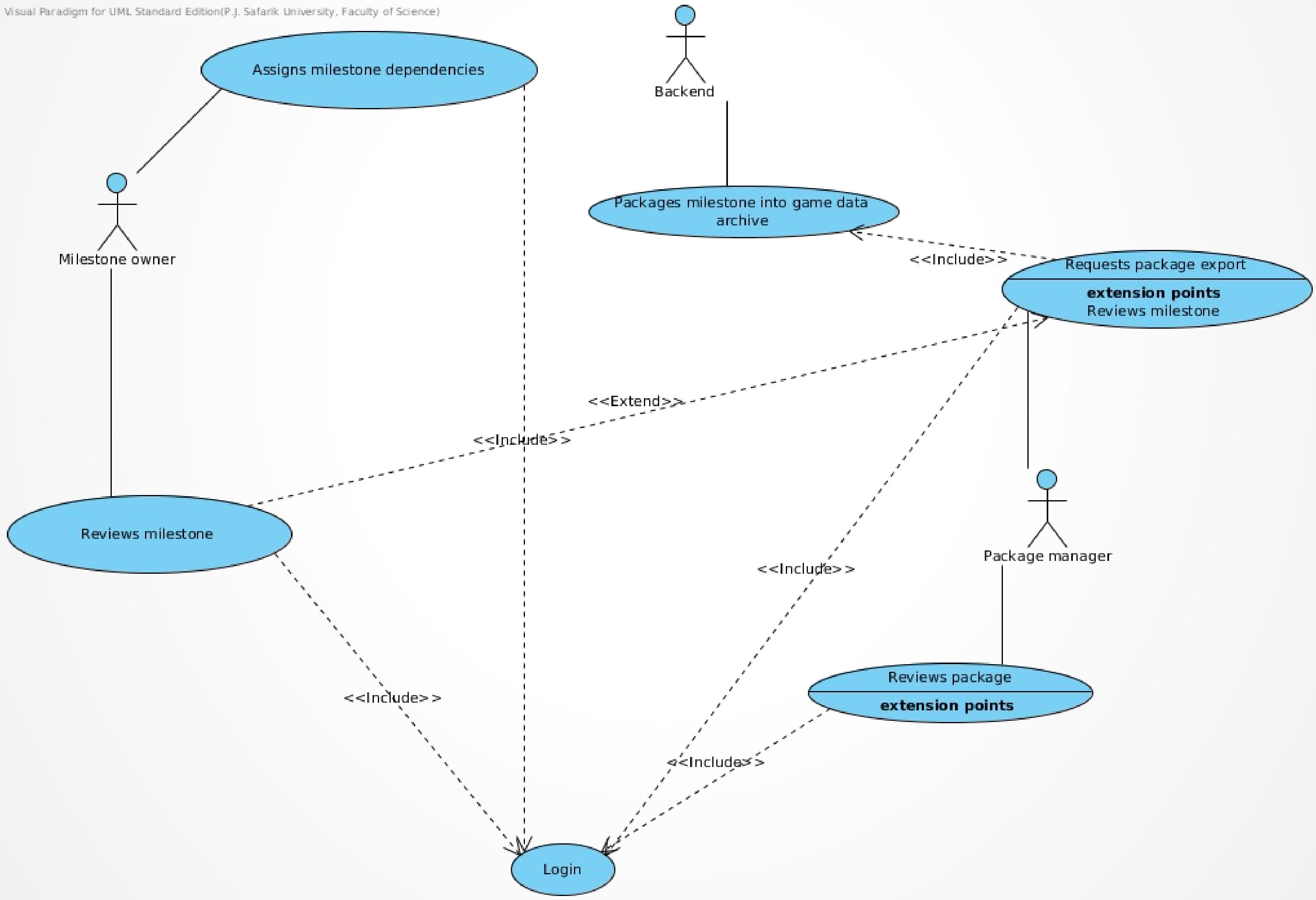
Asset management class diagram



Asset upload use case


Visual Paradigm for UML Standard Edition(P.J. Safarik University, Faculty of Science)
Asset creator
Upload asset
<<Include>>
Login

Milestones/packaging use case



SWOT analysis

- Strengths:
  - Artist-centric
  - Specialized for game assets
- Weaknesses:
  - Lack of legacy support up front (Perforce, SVN)
- Opportunities:
  - Modder access
  - Support based business model
- Threats:
  - Existing project management solutions