History

- started summer term 2004
- interactive application for Plurix
- specially designed for DSM & transactions
Today

- available for two different systems (one more to come)
- multiple distribution mechanism
- web deployment possible
- ca. 18k lines of code
- sophisticated graphics and ui
Structure

- Wissenheim core
- stubs
  - Sun Java
  - Linux / SJC / OSS lib
  - Rainbow / SJC
- TGOS
- graphics (deployment version)
Structure

- Wissenheim core
- stubs
  - Sun Java
  - Linux / SJC / OSS lib
  - Rainbow / SJC
- TGOS
- graphics (deployment version)

apps.wissenheim
- animation
- basic
- chat
- collisiondetection
- controlinterface
- external
- importer
- ki
- physics
- renderengine
- scenegraph
- scenes
- ui
Structure

- Wissenheim core
- stubs
  - Sun Java
  - Linux / SJC / OSS lib
  - Rainbow / SJC
- TGOS
- graphics (deployment version)

Apps
  - wissenheim
  - main
  - graphics
  - tools
  - naming
  - scheduler
  - devices
Structure

- Wissenheim core
- stubs
  - Sun Java
  - Linux / SJC / OSS lib
  - Rainbow / SJC
- TGOS
- graphics (deployment version)

Apps
- wissenheim
  - main
  - graphics
  - tools
  - naming
  - scheduler
  - devices
tgos

Graphic files
Scene

- **Trunk**
  - **leaf**
  - **Mesh 1**
    - **Texture 1**

- **Bike**
  - **tire**
  - **Mesh 2**
    - **Texture 2**
  - **Mesh 3**
    - **Texture 3**

**distributed**
class Node {

    //-- Plurix
    static void getToken() { special.getToken(); } ...

    //-- Linux / OSS
    static void bot() { special.bot(); }
    static void eot() { special.eot(); }
    static void setAllocator(int type) { special.setAllocator(type); } ...

    //-- TGOS
    static void push(Object object) { special.push(object); }
    static void pull(Object object) { special.pull(object); }
    static void sync(Object object) { special.sync(object); } ...

}
• binary code by SJC compiler
• distribution via OSS library
• special stub libraries for Xlib, GL and threading
• reproduction of standard Java utils
• simple memory management by hand
- main tasks correspond to a transaction

- “Plurix Style“

- distinction between local & shared objects

- selection via Node.setAllocator()

- name service still local
Feedback

- consistency on memory basis is difficult
- cascading memory type setting

```java
public Mesh getMesh(long id) {
    if (hash.exists(id)) return hash.get(id);
    file = NameService.get(id);
    Node.setAllocator(SW_LOCAL);
    Mesh mesh = LoadFileAndRegister(file);
    Node.setAllocator(SW_RESTORE);
    return mesh;
}
```

- transactions span more code than necessary
- orthogonal to program flow
Sun Java

- Java 1.6
- JOGL & JOAL
- Applets & WebStart
- distribution on object level via TGOS
- full Java features usable
- rapid development
- best for designers
• marking on object level

• only basic operations
  ➔ push
  ➔ pull
  ➔ sync

• additional events
  ➔ update
  ➔ read
  ➔ write
  ➔ ...

• layered architecture

<table>
<thead>
<tr>
<th>TGOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replication Layer</td>
</tr>
</tbody>
</table>
Folie 17

Hulls

Scene
- String::name: "MyScene"
- Shape::child

Shape
- Vect3D::translation: [1,1,2]
- Vect3D::rotation: [0,90,0]
- Vect3D::scale: [1,1,1]
- Shape::next

Shape
- Vect3D::translation
- Vect3D::rotation
- Vect3D::scale
- Shape::next
Transactions

- cooperative
- classic versioning
- deferred updates

- collision domains
- usable without events
do {
    begin();
    shadowCopy(object1);
    object1.do()
    .
    .
    shadowCopy(object2);
    object2.do()
    .
    .
    if (notOK) {
        abort();
        break;
    }
} while(!commit())
Implementation

- TA block only for write access
- read access always consistent
- one round trip only
- AspectJ for events possible
- “weakend” transactions
Usage

- Only sporadic usage
- possible due to consistent read access
- e.g. adding new avatar

```java
do {
    shadow(scene);
    shadow(avatar);
    scene.add(avatar);
} while(commit());
```

- usage more difficult than standard Plurix style
- consistency based on program flow
My Goal

- Wissenheim as MMORPG
- using TGOS as distribution basis
- client-server approach
- simple implementation
- XtreemOS for providing usable services (e.g. one ip)
- Goal: „Measurement & Promotion“
Network

- User 1 NormalNode
- User 2 NormalNode
- User n NormalNode

Mapper

FATCave
- SceneTask NormalNode
- SuperPeer

Avatars
- SuperPeer

Blobby
- SceneTask NormalNode
- SuperPeer

Global Naming
synergies

- **Pro**
  - scene & data sharing
  - memory based
  - peer-to-peer programming

- **Cons**
  - granularity
  - availability to end user
  - three different protocols
  - object formats (if objects at all)
  - multi-consistency approach
  - availability