SharpLudus

Specifying a Software Factory for 2D Adventure Computer Games Development

André Furtado
awbf@cin.ufpe.br

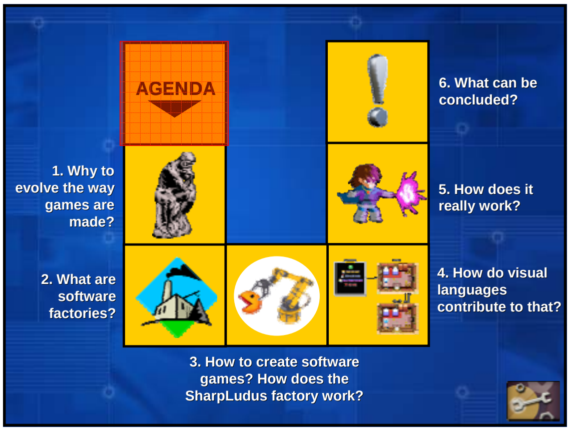
SEC(R), November 2006

Who am I?

- Brazilian Microsoft Student Partner, BS/MSc
- Supported by Microsoft Academic (Brazil) and Microsoft Innovation Center
- From... Recife, Brazil

Sharp... what?

Improving Game Development Experience through Software Factories and Domain-Specific Languages



1. Why to evolve the way games are made?

2. What are software factories?

3. How to create software games? How does the SharpLudus factory work?

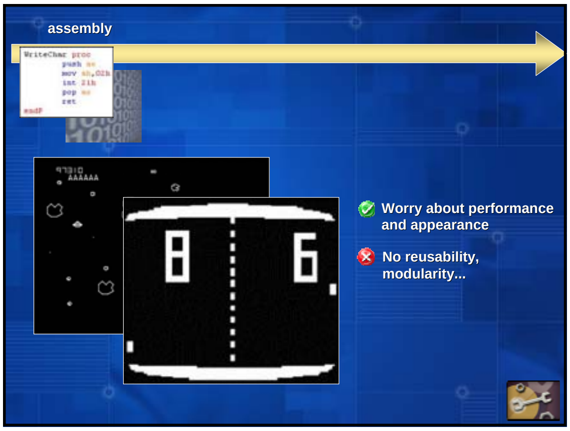
4. How do visual languages contribute to that?

5. How does it really work?

6. What can be concluded?

Game Development

History, deficiencies, opportunities and trends

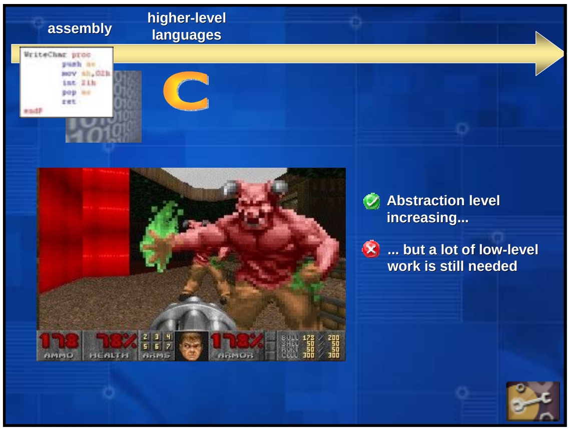


assembly

```
WriteChar proc
          push ax
          mov ah,02h
          int 21h
          pop ax
          ret
endP
```

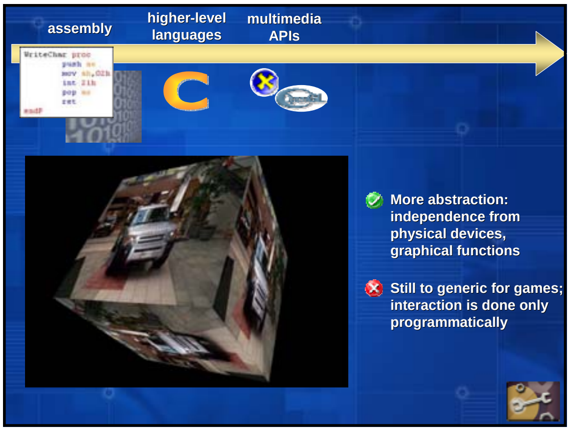
✓ Worry about performance and appearance

✗ No reusability, modularity...



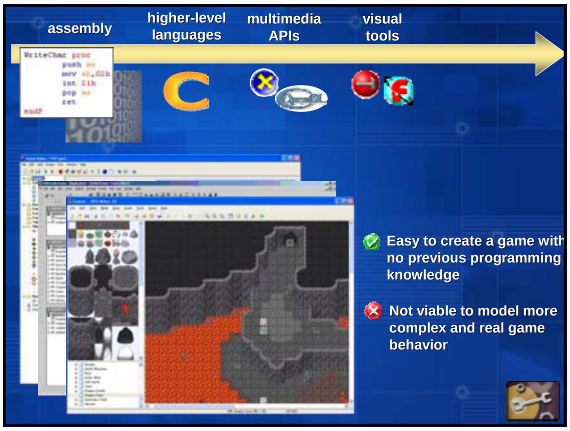
```
WriteChar proc
    push ax
    mov ah,02h
    int 21h
    pop ax
    ret
endP
```

- ✓ Abstraction level increasing...
- ✗ ... but a lot of low-level work is still needed



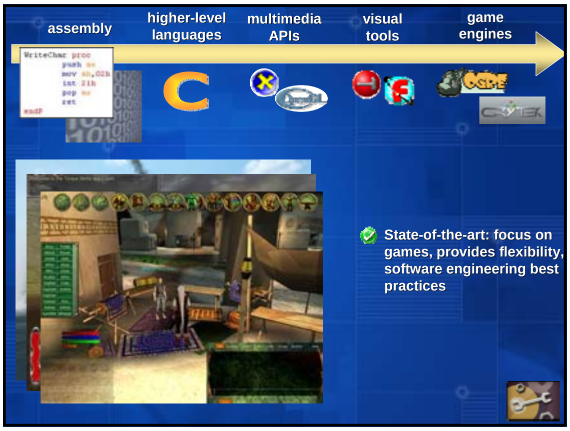
```
WriteChar proc
    push ax
    mov ah,02h
    int 21h
    pop ax
    ret
endP
```

- ✓ More abstraction: independence from physical devices, graphical functions
- ✗ Still to generic for games; interaction is done only programmatically



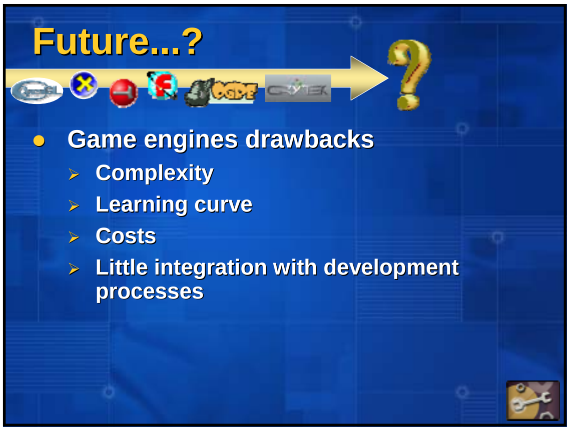
```
WriteChar proc
    push ax
    mov ah,02h
    int 21h
    pop ax
    ret
endP
```

- ✓ Easy to create a game with no previous programming knowledge
- ✗ Not viable to model more complex and real game behavior



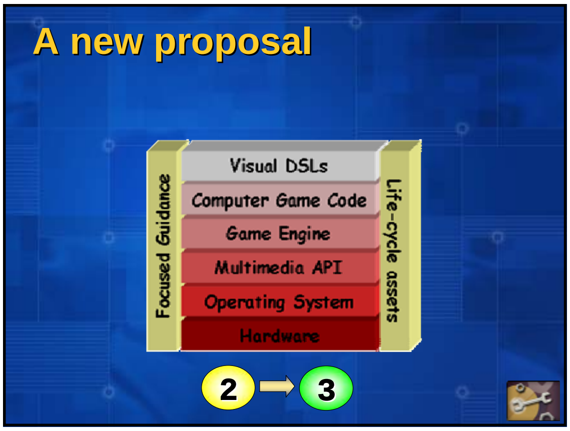
```
WriteChar proc
    push ax
    mov ah,02h
    int 21h
    pop ax
    ret
endP
```

- ✓ State-of-the-art: focus on games, provides flexibility, software engineering best practices



Future...?

- Game engines drawbacks
  - Complexity
  - Learning curve
  - Costs
  - Little integration with development processes



A new proposal

AGENDA

1. Why to evolve the way games are made?

2. What are software factories?

3. How to create software games? How does the SharpLudus factory work?

4. How do visual languages contribute to that?

5. How does it really work?

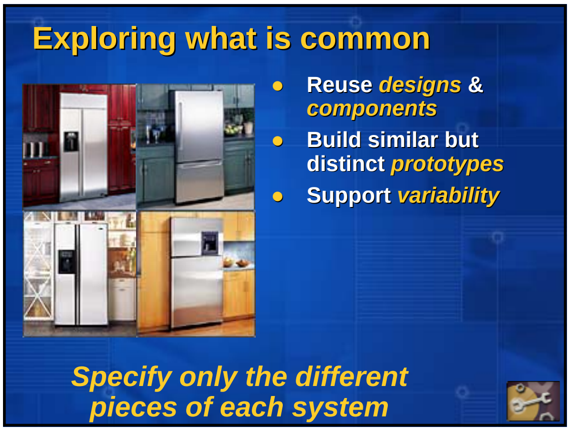
6. What can be concluded?

Software Factories

Motivation, concepts and DSLs

Software Develpment as Craftsmenship

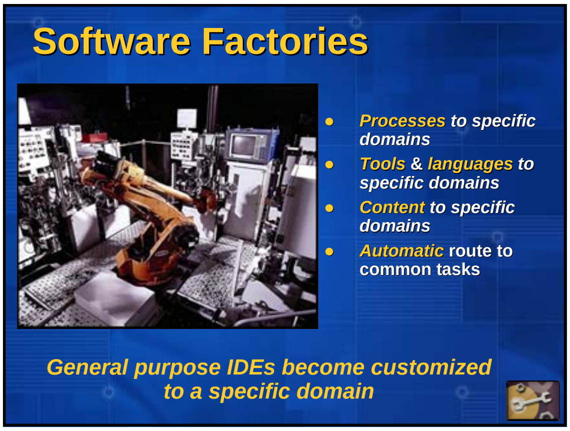
- Intensive work
- Generic tools
- Generic processes
- One application a time
- Hand-made, from scratch
- Little reuse



Exploring what is common

- Reuse *designs* & *components*
- Build similar but distinct *prototypes*
- Support *variability*

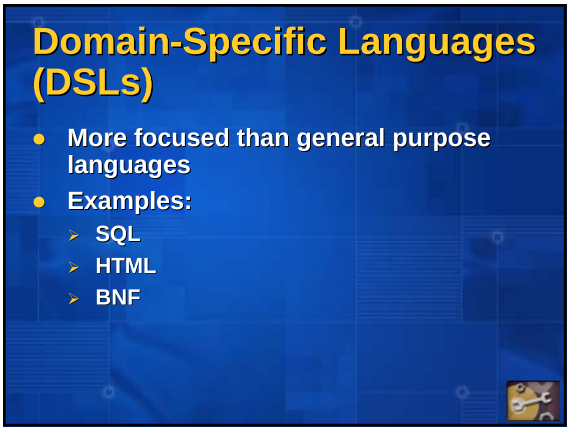
Specify only the different pieces of each system



Software Factories

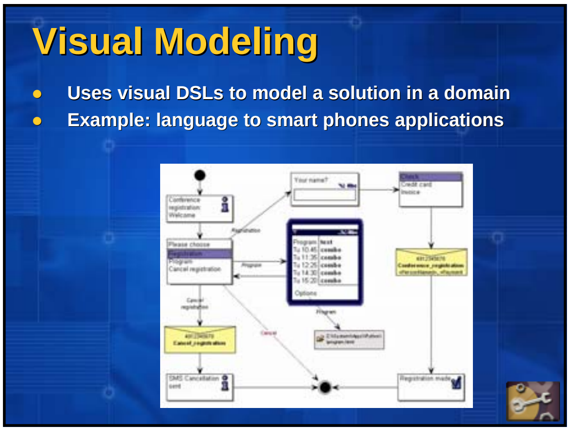
- *Processes to specific domains*
- *Tools & languages to specific domains*
- *Content to specific domains*
- *Automatic* route to common tasks

General purpose IDEs become customized to a specific domain



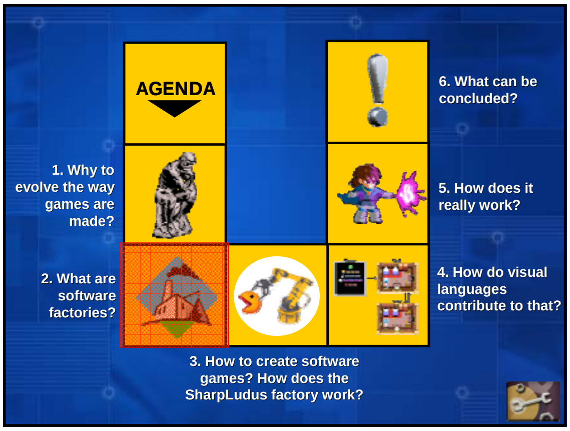
Domain-Specific Languages (DSLs)

- More focused than general purpose languages
- Examples:
  - SQL
  - HTML
  - BNF



Visual Modeling

- Uses visual DSLs to model a solution in a domain
- Example: language to smart phones applications



1. Why to evolve the way games are made?

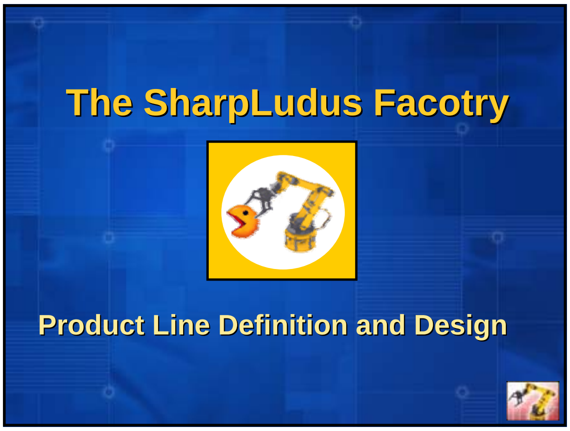
2. What are software factories?

3. How to create software games? How does the SharpLudus factory work?

4. How do visual languages contribute to that?

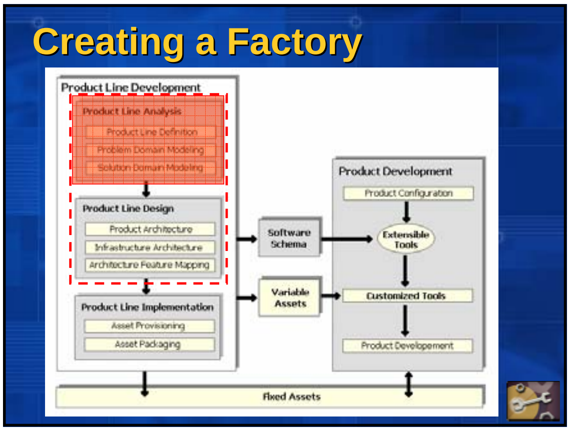
5. How does it really work?

6. What can be concluded?



The SharpLudus Facotry

Product Line Definition and Design



Creating a Factory

Product Line Development

Product Line Analysis

Product Line Definition

Problem Domain Modeling

Solution Domain Modeling

Product Line Design

Product Architecture

Infrastructure Architecture

Architecture Feature Mapping

Product Line Implementation

Asset Provisioning

Asset Packaging

Software Schema

Variable Assets

Product Development

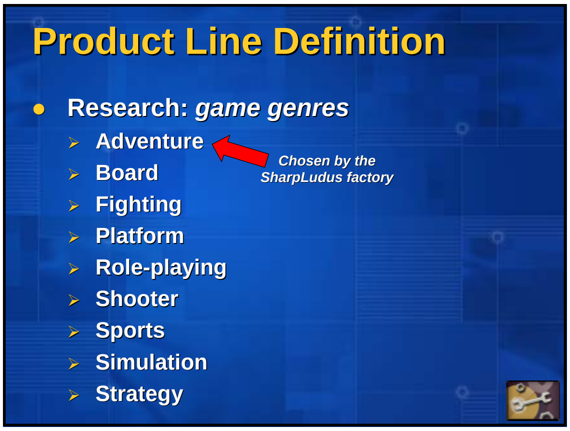
Product Configuration

Extensible Tools

Customized Tools

Product Developement

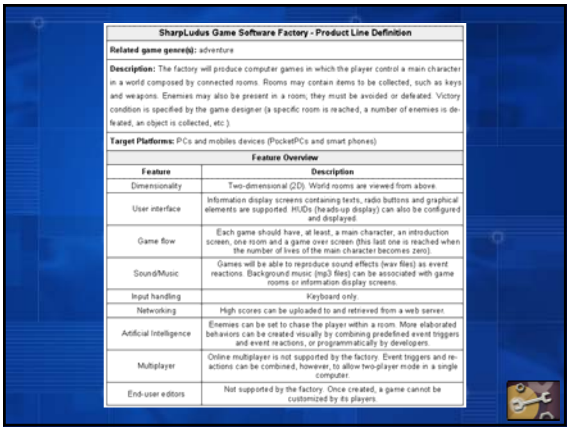
Fixed Assets



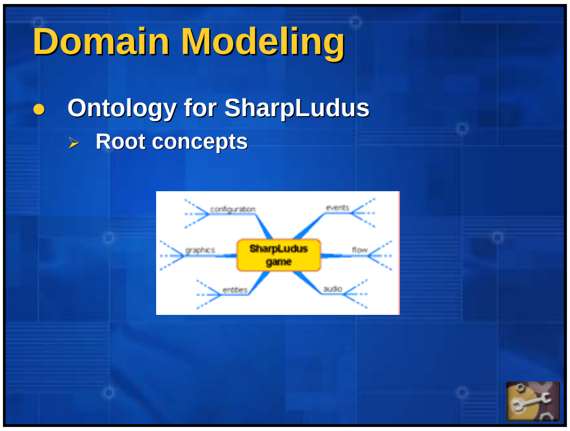
Product Line Definition

- Research: *game genres*
  - Adventure
  - Board
  - Fighting
  - Platform
  - Role-playing
  - Shooter
  - Sports
  - Simulation
  - Strategy

Chosen by the SharpLudus factory

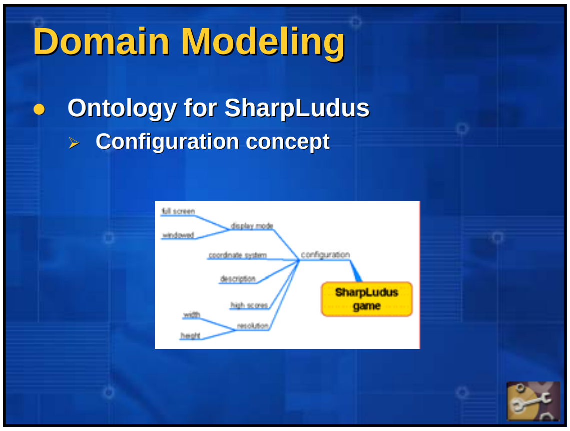


SharpLudus Game Software Factory - Product Line Definition	
Related game genre(s): adventure	
Description: The factory will produce computer games in which the player control a main character in a world composed by connected rooms. Rooms may contain items to be collected, such as keys and weapons. Enemies may also be present in a room; they must be avoided or defeated. Victory condition is specified by the game designer (a specific room is reached, a number of enemies is defeated, an object is collected, etc.)	
Target Platforms: PCs and mobiles devices (PocketPCs and smart phones)	
Feature Overview	
Feature	**Description**
Dimensionality	Two-dimensional (2D). World rooms are viewed from above.
User interface	Information display screens containing texts, radio buttons and graphical elements are supported. HUDs (heads-up display) can also be configured and displayed.
Game flow	Each game should have, at least, a main character, an introduction screen, one room and a game over screen (this last one is reached when the number of lives of the main character becomes zero).
Sound/Music	Games will be able to reproduce sound effects (wav files) as event reactions. Background music (mp3 files) can be associated with game rooms or information display screens.
Input handling	Keyboard only.
Networking	High scores can be uploaded to and retrieved from a web server.
Artificial Intelligence	Enemies can be set to chase the player within a room. More elaborated behaviors can be created visually by combining predefined event triggers and event reactions, or programmatically by developers.
Multiplayer	Online multiplayer is not supported by the factory. Event triggers and reactions can be combined, however, to allow two-player mode in a single computer.
End-user editors	Not supported by the factory. Once created, a game cannot be customized by its players.



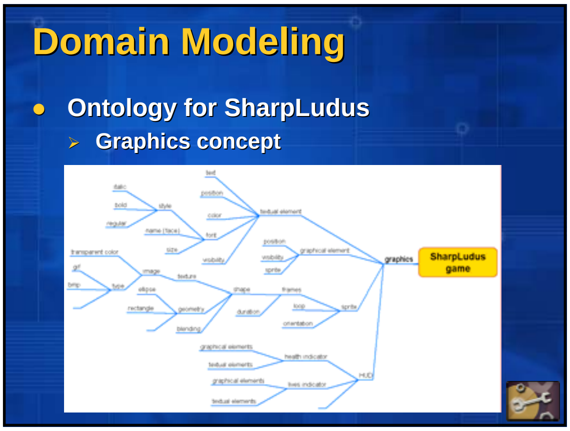
Domain Modeling

- Ontology for SharpLudus
  - Root concepts



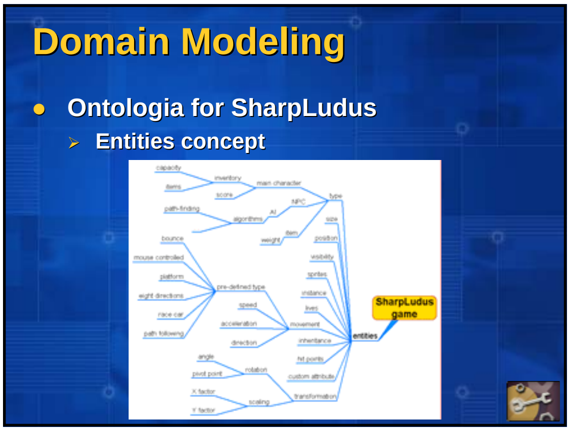
Domain Modeling

- Ontology for SharpLudus
  - Configuration concept



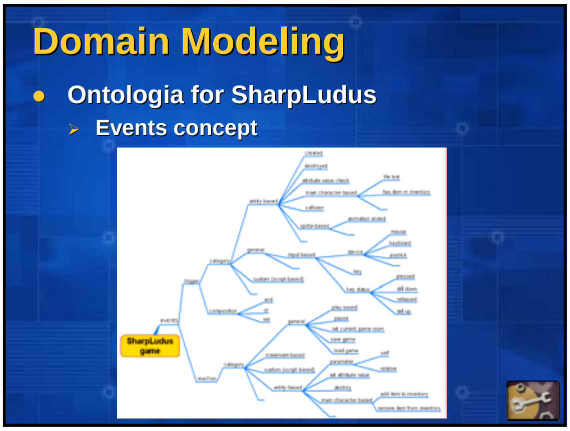
Domain Modeling

- Ontology for SharpLudus
  - Graphics concept



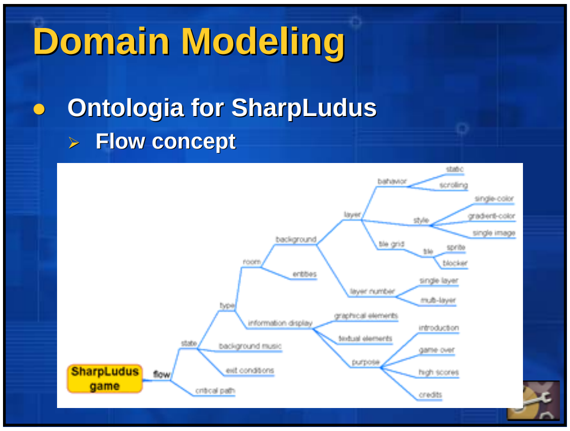
Domain Modeling

- Ontologia for SharpLudus
  - Entities concept



Domain Modeling

- Ontologia for SharpLudus
  - Events concept



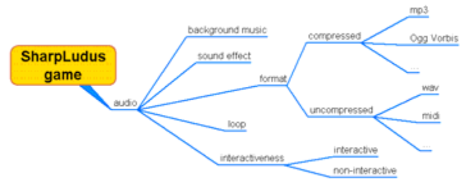
Domain Modeling

- Ontologia for SharpLudus
  - Flow concept

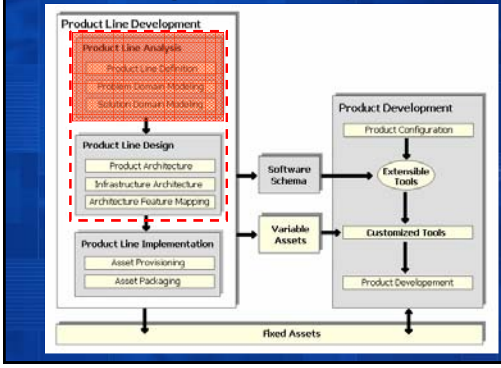
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Domain Modeling

- Ontologia for SharpLudus
  - Audio concept



Creating a Factory

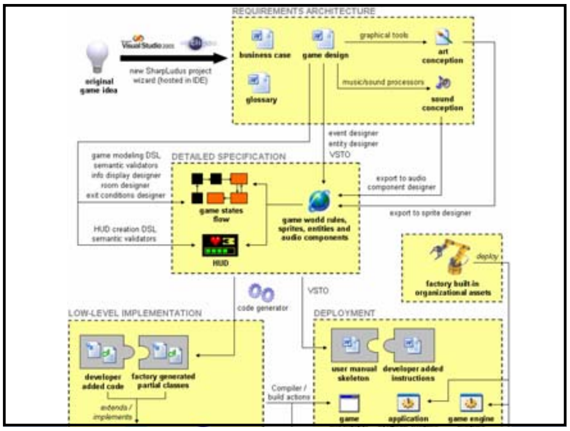


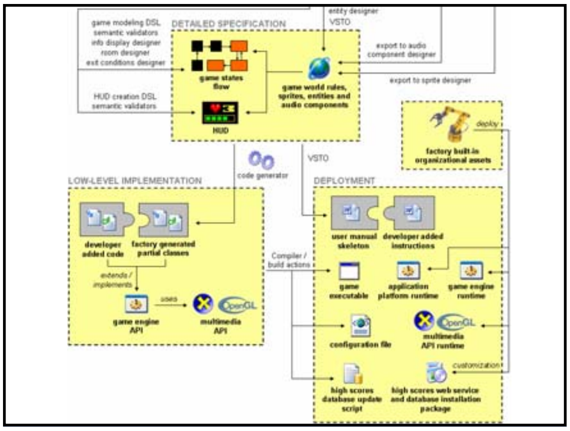
Product Line Design

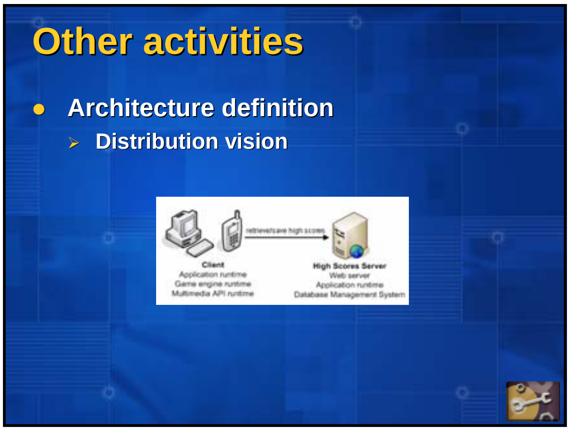
- Goal: to define...
  - Factory specification
  - Architecture
  - Processes
  - Map requirements variation into factory assets

Product Line Design

- Factory schema

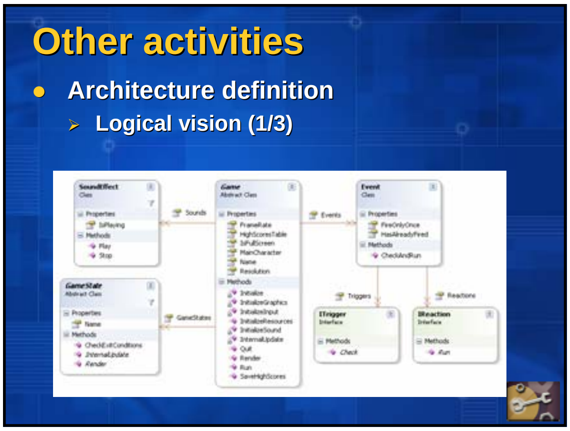






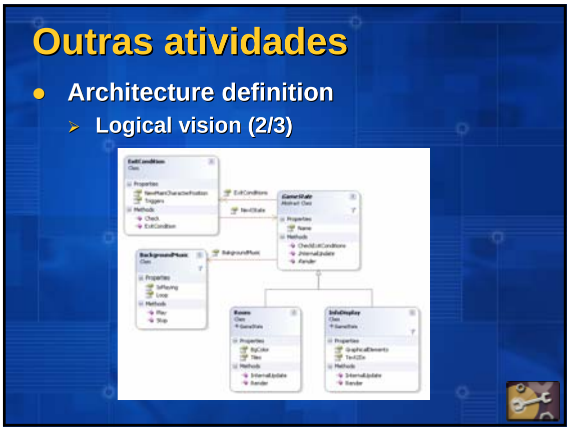
Other activities

- Architecture definition
  - Distribution vision



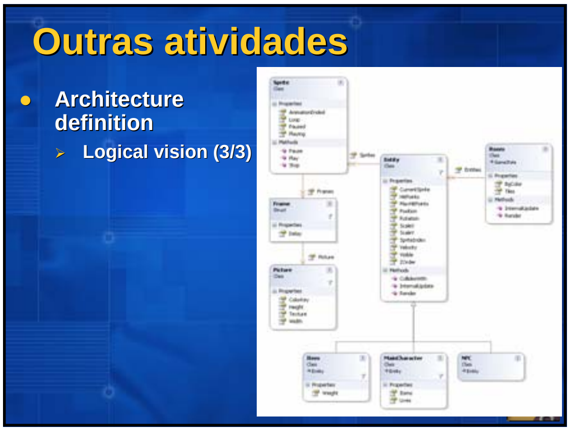
Other activities

- Architecture definition
  - Logical vision (1/3)



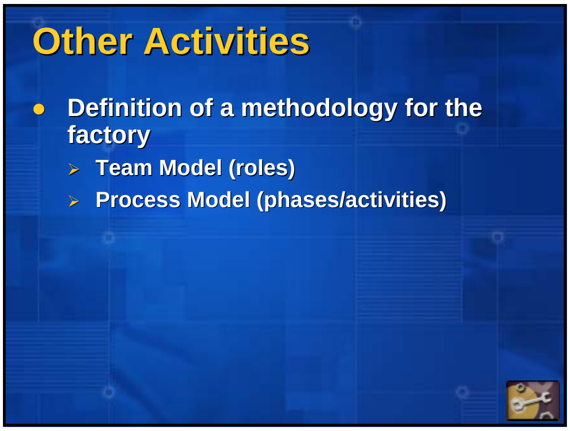
Outras atividades

- Architecture definition
  - Logical vision (2/3)



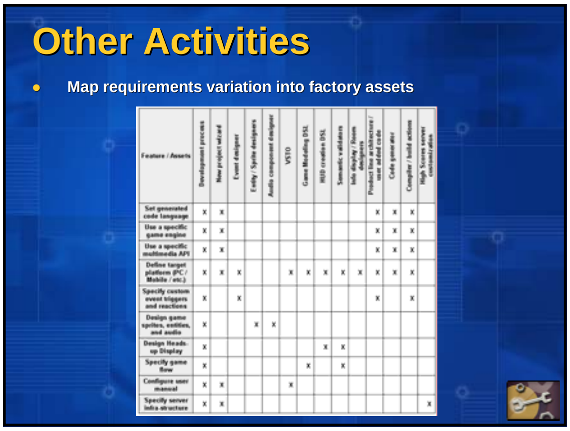
Outras atividades

- Architecture definition
  - Logical vision (3/3)



Other Activities

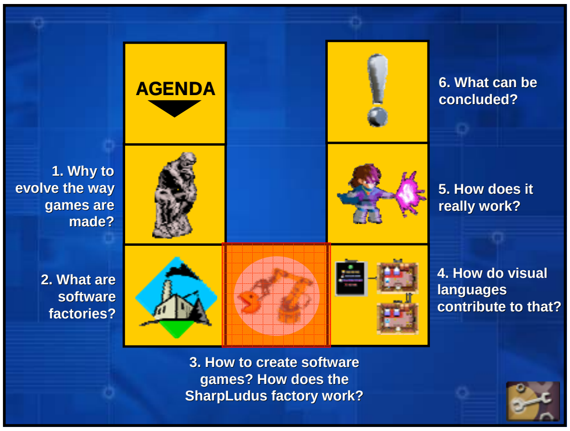
- Definition of a methodology for the factory
  - Team Model (roles)
  - Process Model (phases/activities)



Other Activities

- Map requirements variation into factory assets

Feature / Assets	Development process	New project wizard	Event designer	Entity / Sprite designers	Audio component designer	VSTO	Game Modeling DSL	HUD creation DSL	Semantic validators	Info display / Room designers	Product line architecture / user added code	Code generator	Compiler / build actions	High Scores server customization
Set generated code language	x	x									x	x	x	
Use a specific game engine	x	x									x	x	x	
Use a specific multimedia API	x	x									x	x	x	
Define target platform (PC / Mobile / etc.)	x	x	x			x	x	x	x	x	x	x	x	
Specify custom event triggers and reactions	x		x								x		x	
Design game sprites, entities, and audio	x			x	x									
Design Heads-up Display	x							x	x					
Specify game flow	x						x		x					
Configure user manual	x	x				x								
Specify server infra-structure	x	x												x



AGENDA

1. Why to evolve the way games are made?

2. What are software factories?

3. How to create software games? How does the SharpLudus factory work?

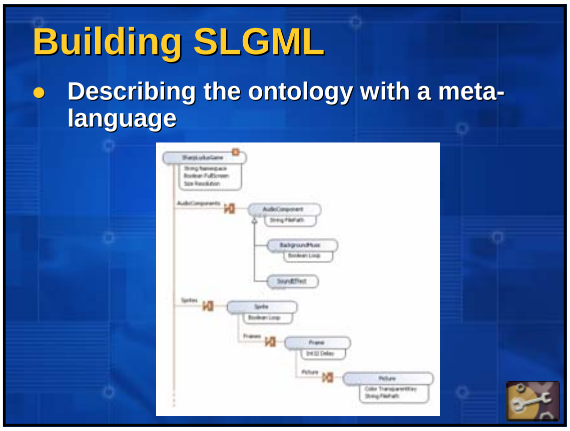
4. How do visual languages contribute to that?

5. How does it really work?

6. What can be concluded?

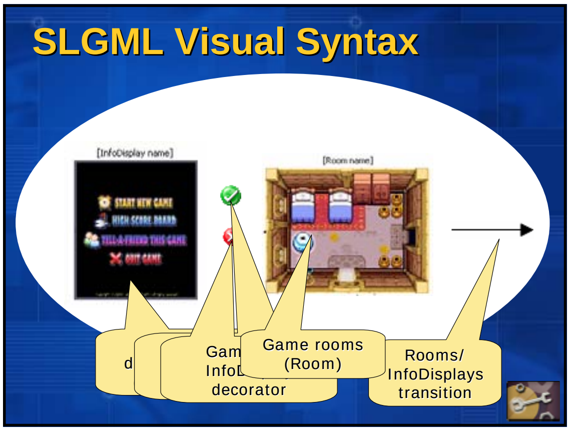
SharpLudus Game Modeling Language (SLGML)

Language specification and related assets

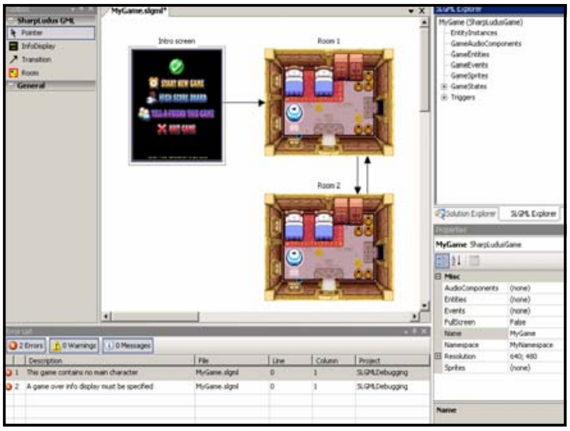


Building SLGML

- Describing the ontology with a meta-language



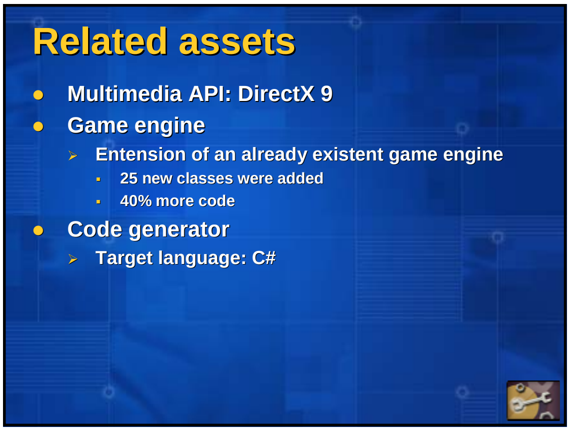
SLGML Visual Syntax



	Description	File	Line	Column	Project
1	This game contains no main character	MyGame.slgml	0	1	SLGMLDebugging
2	A game over info display must be specified	MyGame.slgml	0	1	SLGMLDebugging

AudioComponents	(none)
Entities	(none)
Events	(none)
FullScreen	False
Name	MyGame
Namespace	MyNamespace
Resolution	640; 480
Sprites	(none)

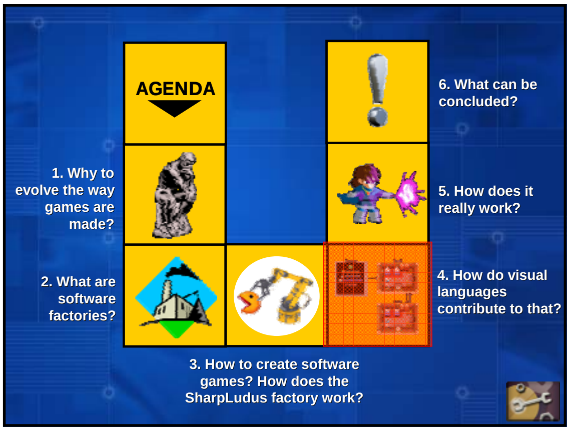
Name



Related assets

- Multimedia API: DirectX 9
- Game engine
  - Entension of an already existent game engine
    - 25 new classes were added
    - 40% more code
- Code generator
  - Target language: C#

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.



AGENDA

1. Why to evolve the way games are made?

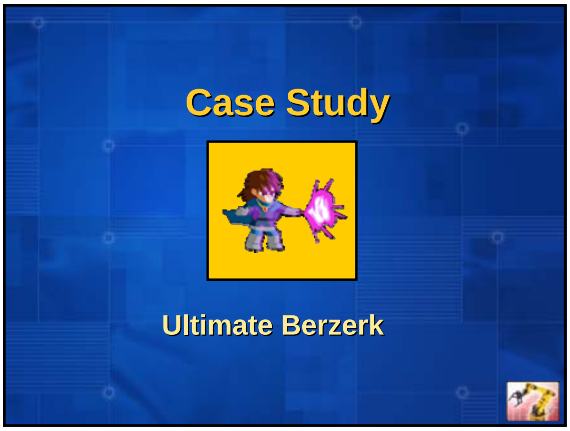
2. What are software factories?

3. How to create software games? How does the SharpLudus factory work?

4. How do visual languages contribute to that?

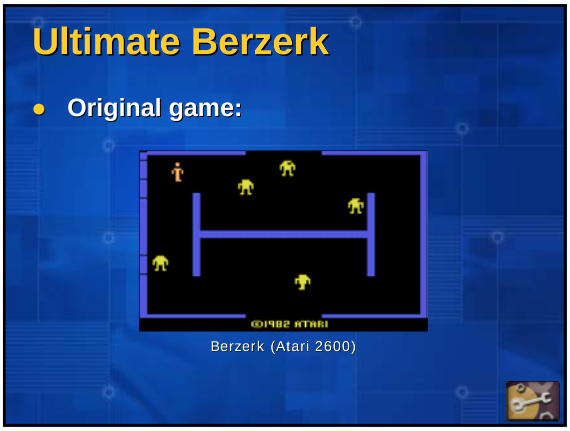
5. How does it really work?

6. What can be concluded?



Case Study

Ultimate Berzerk



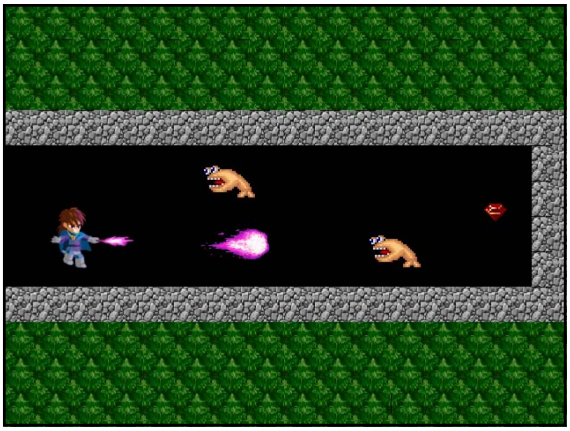
Ultimate Berzerk

- Original game:

Berzerk (Atari 2600)

Ultimate Berzerk

- New features:
  - Different types of enemies
  - Items to be collected
  - Sound effects and music
  - ...



Ultimate Berzerk

- Game design:

John needs to get the diamond and find the exit

he can launch fireballs in enemies if he gets the Weapon

Ultimate Berzerk

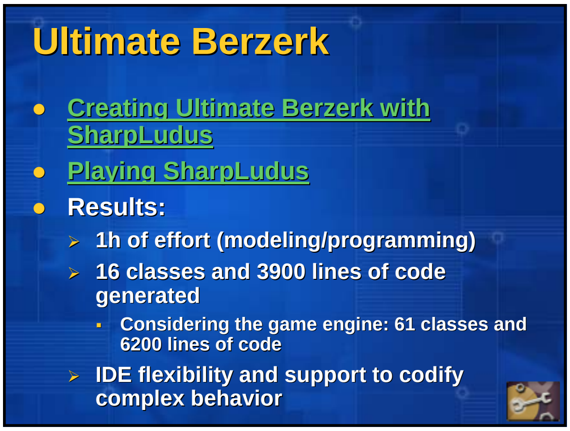
- Enemies

static

chaser

launches a fireball each 3 seconds

has a pre-defined movement



Ultimate Berzerk

- Creating Ultimate Berzerk with SharpLudus
- Playing SharpLudus
- Results:
  - 1h of effort (modeling/programming)
  - 16 classes and 3900 lines of code generated
    - Considering the game engine: 61 classes and 6200 lines of code
  - IDE flexibility and support to codify complex behavior

AGENDA

1. Why to evolve the way games are made?

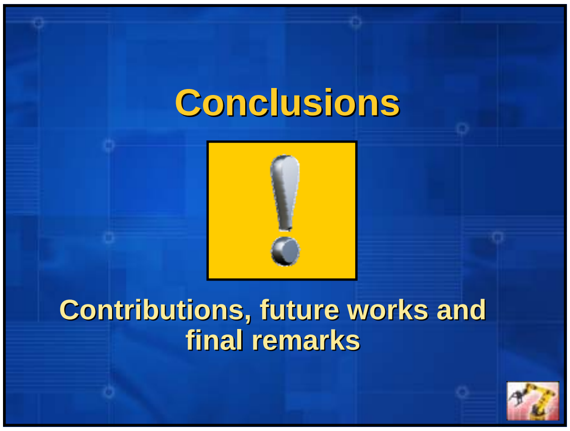
2. What are software factories?

3. How to create software games? How does the SharpLudus factory work?

4. How do visual languages contribute to that?

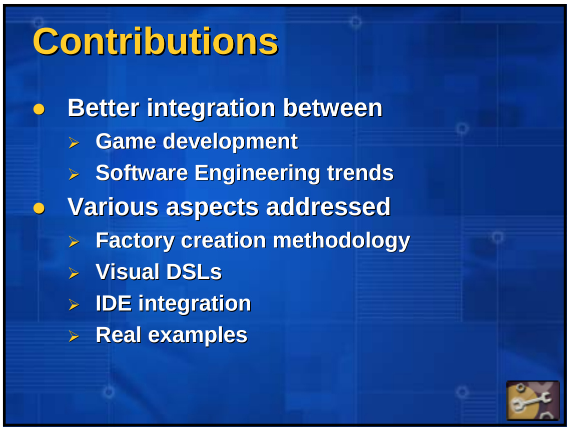
5. How does it really work?

6. What can be concluded?



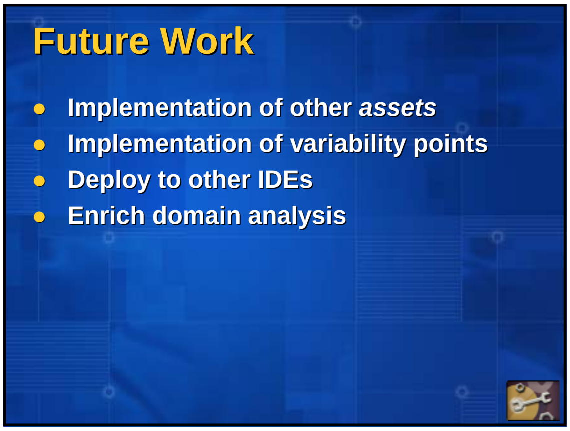
Conclusions

Contributions, future works and final remarks



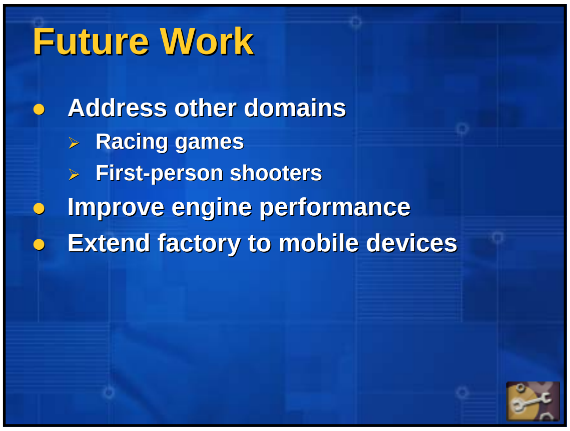
Contributions

- Better integration between
  - Game development
  - Software Engineering trends
- Various aspects addressed
  - Factory creation methodology
  - Visual DSLs
  - IDE integration
  - Real examples



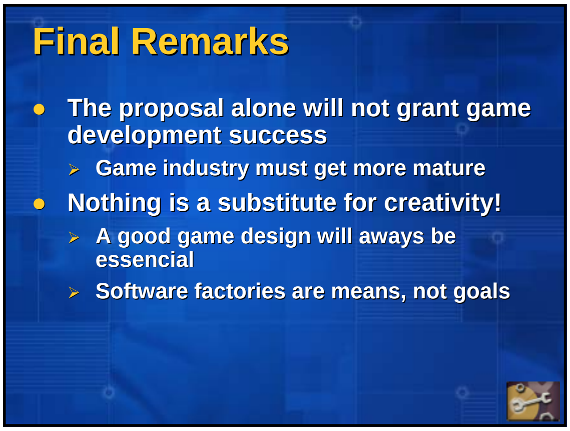
Future Work

- Implementation of other *assets*
- Implementation of variability points
- Deploy to other IDEs
- Enrich domain analysis



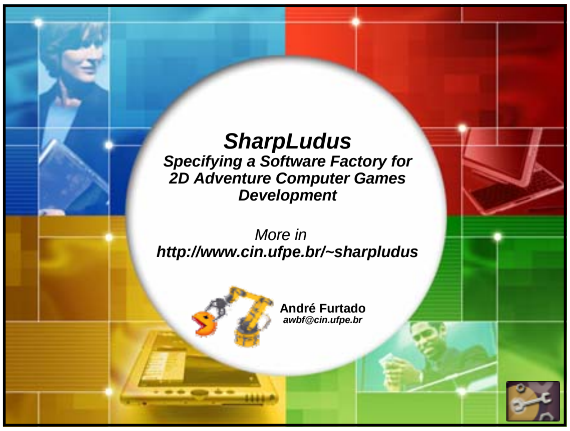
Future Work

- Address other domains
  - Racing games
  - First-person shooters
- Improve engine performance
- Extend factory to mobile devices



Final Remarks

- The proposal alone will not grant game development success
  - Game industry must get more mature
- Nothing is a substitute for creativity!
  - A good game design will aways be essencial
  - Software factories are means, not goals



SharpLudus

Specifying a Software Factory for 2D Adventure Computer Games Development

More in
http://www.cin.ufpe.br/~sharpludus

André Furtado
awbf@cin.ufpe.br

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.