Game Developers Conference™ Europe 2011
August 15-17, 2011 | Cologne, Germany
www.GDCEurope.com

A massive challenge: The cross-platform approach of the mobile MMO TibiaME

Benjamin Zuckerer
Product Manager, CipSoft GmbH

What is this session about?

- Introduction to CipSoft and TibiaME
- TibiaME's cross platform approach
  - Architecture
  - Development costs
  - Tool chain
- Technical challenges
- UI Design challenges
- Payment
- Testing
- Lessons learnt

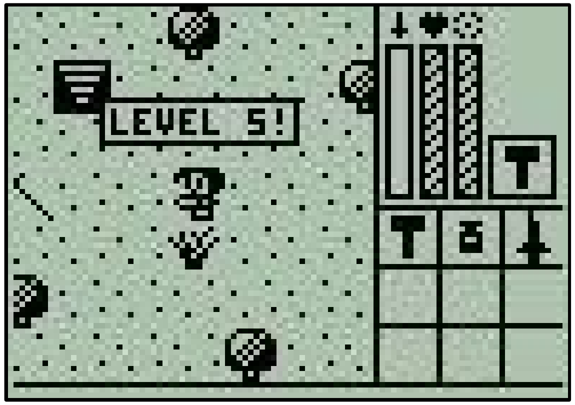
Company

- CipSoft GmbH
- founded 2001
- independent developer
- 4 owners
- 62 employees
- innovative online games

Tibia TibiaME Fiction Fighters

TibiaME

- 2D fantasy MMORPG for mobile devices
- Online since May 2003
- Business model
  - Free2play with optional subscriptions
  - No microtransactions (yet)
- Available for various platforms
  - J2ME, Symbian Series60
  - Now also for: iOS, Android and Web



Classic client

- Designed for screens with 128x128 or 176x208
- 20x20 px graphic assests, walking animations
- No sound or music
- Only keyboard / joystick based controls
- Available for J2ME and Symbian Series 60

ExtremeClient

- We wanted to
  - Use the same protocol (for server communication)
  - Support Touchscreen devices (Symbian & iOS)
  - Take advantage of the higher screen resolutions
  - Have the same chance to succeed in the game, no matter on which device TibiaME is played

→ 2D Retro graphics

→ start from scratch, new client 'ExtremeClient'

Why go cross-platform?

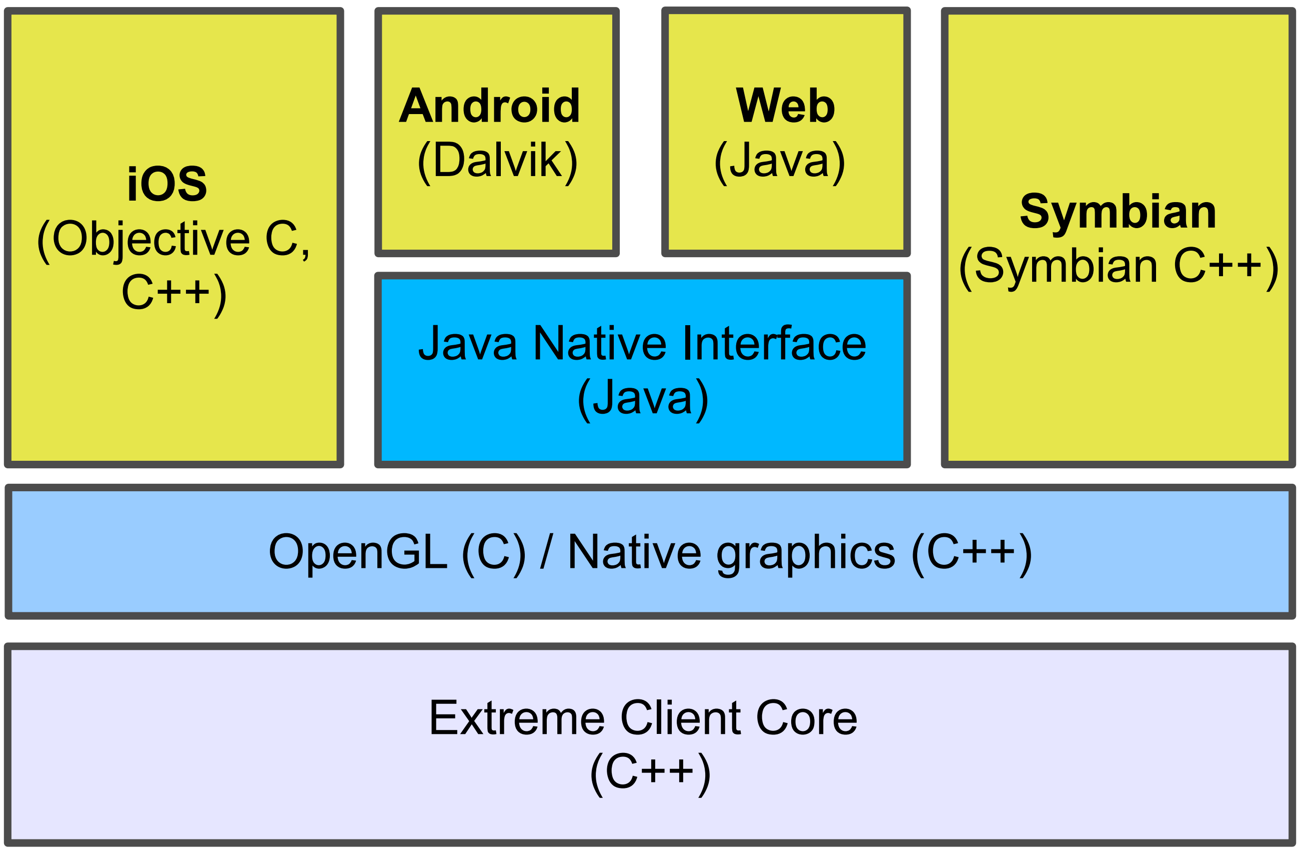
CipSoft online entertainment

- Fast changing market
- Be flexible, don't bet on a single Apple
- Intense competition – spread your risks
- Early bird advantage on new platforms
- More platforms → more potential customers
- Lower development costs on the long run

- Cross-platform approach
  - Started with Symbian and iOS

    → Added Android client and Web client later on

Architecture



- Input
- System dialogs
- Device characteristics

- Graphic output

- Data
- Communication
- Layout

Development costs

- Core code makes up ~80%
  - So, there is ~20% specific code per platform
  - New features and bugfixes available to all platforms

- Development time
  - Core system + iOS + Symbian (2 Programmers, ~1.5 years)
  - Android + solution for fragmentation, e.g. screen sizes (1 Programmer, ~4 month)
  - Web (1 Programmer, ~2 month)

Tool chain

- Development:
  - Mac / XCode: required for iOS
  - Eclipse (with plugins): all other platforms

- We are using the following libraries:
  - OpenGL ES 1.1
  - OpenAL
  - Libpng
  - RapidXML

Tool chain

- Graphic and sound assets
  - ImageMagick to scale graphics
  - Our own tools for texture packaging
  - ffmpeg and lame to convert sound and music

- Localisation
  - 'xliff' file format
  - Pootle
    - open source translation tool
    - Web based and easy to use

Technical challenges

- Target devices
  - Does your game require certain hardware specs?
  - Any features only available for certain OS versions?
  - Which platforms do you want to support?

  → Make as many features optional as possible

  → Test your assumptions with a prototype

Technical challenges

CipSoft online entertainment

- Fragmentation
  - CPU speed, GPU speed, available memory
  - Different OS versions → available features
  - OpenGL ES 1.1 not mandatory on many Android versions
    - Requires workarounds for OpenGL ES 1.0
  - Not even bugs are consistent across devices
  - Many different screen sizes
    - QVGA (240x320) up to Retina displays (640x960)
    - Different aspect ratios
    - Can't scale the pixel based graphics + need different UIs

Solutions

- Precreated assets
  - Sound & music assets created uncompressed
    - Then converted to all formats required by the devices
    - Clients are packaged with the correct format
  - All graphic assets are drawn in 64x64 px
    - Then graphic assets are prescaled to each possible size between 35x35 and 99x99 px
    - At first launch client will calculate the required size and then download the best fitting assets and if not packaged, the UI

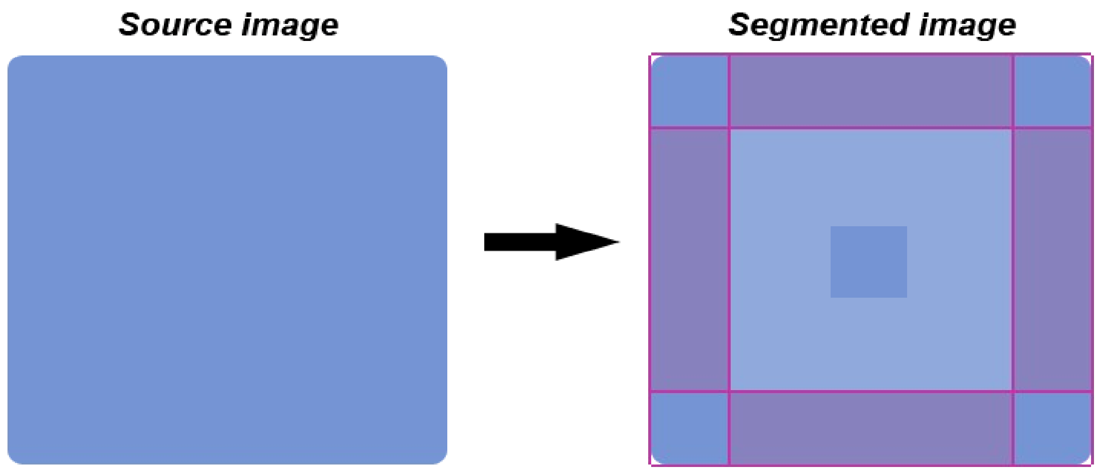
→ Problem: Download size vs connection speed

Solutions

- Flexible UI system
  - Currently 3 UIs, covering the common aspect ratios
  - Possibilty to add new UIs (e.g. Tablet UI)
  - XML description file (using filters)
    - Defines the layout of the UI-elements in the different UIs
  - Filters used to:
    - activate UI-elements, depending on UI or screen orientation
    - take care of OS specific design differences
    - consider the device input capability (Touchscreen, Keyboard or combination)

Solutions

- Flexible UI system
  - Portrait and landscape support
    - Reuses as many portrait graphics as possible
    - Requires a few additional graphics for landscape
  - Using 9-grid and patterns for UI where possible



  - If not possible the UI element is offered in different sizes
    - Original size, 125%, 150% and 200%

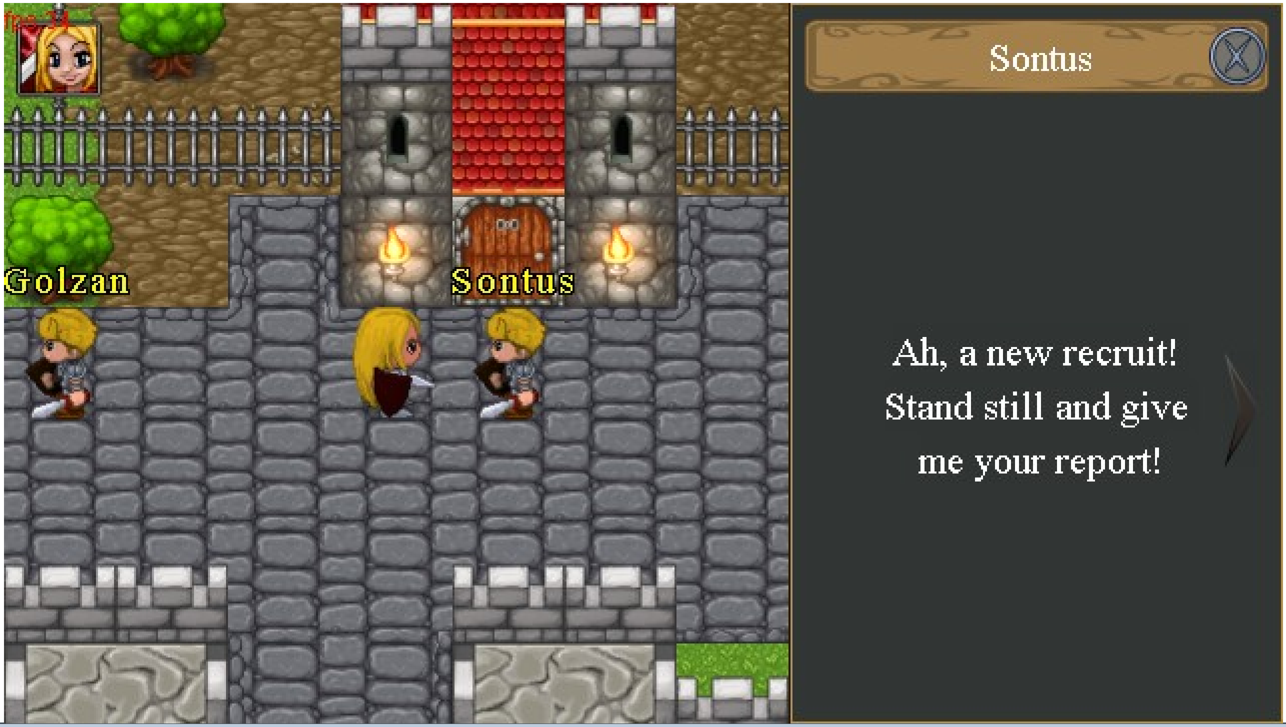
Different UIs: Portrait

Grid: 7 wide, 9 high
Keyboard & Joystick
(240x320)

Grid: 7 wide 10 high
Touchscreen
(320x480)

Grid: 7 wide, 11 high
Touchscreen & Keyboard
(360x640)

Different UIs: Landscape



UI Design challenges

- Classic game is really old
  - Designed around the phone's numberpad and joystick
  - Dialogs also optimized for these controls
  - Screen only used to show information

- Touchscreen support
  - Completely new controls for movement
  - Virtual keyboard requires space, covers a lot of screen
  - Dialog structure needs to work with Key & Touch

→ Prototype to get the controls right

UI Design challenges

- Different screen types
  - Capacitive & resistive screens

    → Don't use fancy touch gestures (e.g. Drag & Drop)

    → Get the details right, e.g. size of the scroll bar

- Variety of input
  - Touchscreen, Keyboard or combination of both

    → Plan for different controls:

    - Touch (simple touch, long touch,..)
    - Virtual joystick
    - Keyboard, Xperia Play

UI Design challenges

- OS specific design guides
  - Symbian
    - switched 'OK' and 'CANCEL' buttons
    - needs to work with a stylus
  - iOS Human Interface Guidelines
    - size of tappable UI elements is 44 x 44 points
  - Android Menu Design Guidelines

→ Try to keep the look and feel across platforms

→ Solved via filters in XML description

Payment

- Payment on different platforms
  - Apple will not allow links or other payment options in the game – Only Apple InApp purchase!
  - Google doesn't care! Here you can offer other payment methods along with InApp purchase

- Payment server
  - More secure - prevents most hacking scenarios
  - Checks if customer purchases really are valid
  - Allows us to offer multiple payments methods to our customers, depending on country and used device

Testing

- Going cross-platform also means more testing
  - Bugs in the core can effect all platforms
  - Changes in the UI can also have unwanted effects
- Fragmentation requires even more testing

  → Buy devices for testing on all platforms

Distribution channels

- Many different application stores
  - AppStore (iOS)
  - MarketPlace (Android)
  - Ovi (Symbian, J2ME)
  - GetJar (Symbian, J2ME, Android, ...)

- Difficult to update all clients at a fixed date
  - → Plan for long review times or even rejections
  - → Update mentality differs across platforms
  - → Support older client versions

Lessons learnt

- You need to make compromises for cross-platform
  - Depending on your target devices
  - Depending on the type of controls you want to support

- 3D would have solved most scaling problems
  - Easier to create a tablet optimized version of the game
  - UI scaling and layout still a challenge

Lessons learnt

- Review process
  - Apple's review process takes long, especially when you have a bug that needs fixing!
  - No Beta testing possible with iOS, but great on Android
  - Iterate on Android and then submit to iOS
  - Client update problem: How can Apple test a new version, if the productive system is not updated yet?

    → Redirect newer clients to a special test enviroment

Lessons learnt: iOS

- Pushing the App into the background causes problems with online games (network traffic)

  → Close the connection – open a new one on start
- Test with the newest available iOS version
- Don't trust the simulator – always test on a real device
- Check if you really need to support older iOS versions, as upgrade mentality is good

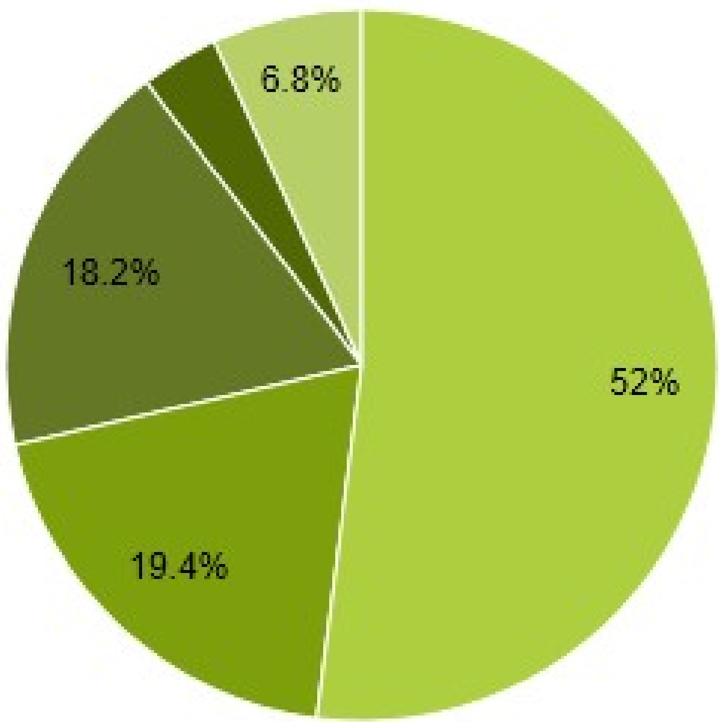
Lessons learnt: Android

- Pushing the App into the background causes problems with network traffic and OpenGL context is lost

  → Close the complete client, offer a quickstart option
- '.nomedia' file for Android (or your pics and music turn up in the media player app on some devices)
- OpenGL implementations vary (driver)
- NDK makes debugging difficult
- Android 1.6 and 2.1: InApp purchases buggy
- Plan for extra testing time

Lessons learnt: Android

- Support for older versions → a lot of work!
- Android OS distribution in TibiaME



Android 3.01	0.2 %
Android 2.3.3	19.4 %
Android 2.3	2.8 %
Android 2.2	52.0 %
Android 2.1	18.2 %
Android 1.6	3.6 %

Conclusions

- Plan on a project basis
  - Choose your platforms
  - Symbian has been discontinued
- Make as many features optional as possible
- Solve the screen size problem
- Prototype and test
  - Required performance
  - If your controls work
- Plan for extra testing time
- Use the different distribution channels

Thanks!

- zuckerer@cipsoft.com
- www.cipsoft.com

Inside Tibia -
The Technical Infrastructure of an MMORPG

Wednesday 12:40- 1:30
Offenbachsaal, 1st Level

We're hiring!
at gamescom:
"Jobs & Karriere"
hall 8 booth B40
http://www.cipsoft.com/jobs