# Sergii Mikhtoniuk

| | |
|---|---|
| **Phone:** | +38 098 934 1820 |
| **Location:** | Kiev, Ukraine |
| **E-Mail:** | mikhtonyuk@gmail.com |
| **Skype:** | mikhtonyuk.sergey |
| **GitHub:** | github.com/mikhtonyuk |

## SUMMARY

Professional software engineer with development experience in a variety of languages and technologies. Participated in development of core game engine components, in design and implementation of server backend of award-winning online game being played by millions of people across the world. Experienced in development workflows, supervision of distributed teams, integration with publishers' infrastructures, and other technical and organizational challenges on a way from production up to launch on multiple territories.

### SPECIALTIES

- APIs and reusable component design
- Parallel programming
- Performance-critical systems
- Large-scale software architecture
- Databases and scalability
- MMO backend development
- Plugin-based extensible architectures
- Test-driven development

### INTERESTS

- Service-oriented architectures
- Distributed systems and algorithms
- Big data, data warehousing, ETL
- Data mining, machine learning
- Database internals
- Compilers and domain-specific languages
- Virtualization and cloud computing
- Functional programming

## EDUCATION & SCIENTIFIC ACTIVITY

Master of Computer Science
2005–2010 Kharkov National University of Radio Electronics, Ukraine

- Graduated with honors
- Government scholarship award
- Top faculty student award
- Member of "Design & Test" students research lab
- 5 awards in software engineering and hardware design contests
- 6 published papers
- External reviewer for 5th International Conference on Embedded and Multimedia Computing (EMC-10)
- Gave lectures on software architecture and computer graphics for students

## TECHNICAL SKILLS

### PROGRAMMING LANGUAGES:

- C++, C#, Java, Erlang, Scala
- Python, Lua, PHP
- Bash, SQL, HLSL / Cg
- XML, Regular expressions
- Assembly, SSE

### WORKFLOW:

- Agile, SCRUM
- Estimation, release planning
- Branching models and multi-SKU codebases
- Technical documentation
- Continuous integration
- Build automation, environment enrollment

### DESIGN:

- Object-oriented analysis and design
- Design patterns, UML
- Reusable API design
- Managing complexity
- Service-oriented architecture
- Plug-in and component-based architectures
- Game engine architecture
- MMO server architecture

### DEVELOPMENT:

- Test-driven development
- Automated functional and stress testing
- Algorithmic and low-level optimization
- Parallel algorithms, concurrency
- Development of domain-specific languages
- Hand-written compilers and compiler generation tools (ANTLR, LEX\YACC family)

**NETWORKING:**
- Low-level protocols
- Messaging protocols (XMPP, AMQP, MQTT)
- Security, counter-hacking
- QoS, load balancing, fault tolerance
- NAT solutions, P2P, host migration
- Scalability and high availability
- Web services SOAP/REST
- XML Schema design

**COMPUTER GRAPHICS:**
- Lighting algorithms and techniques
- Cg\HLSL shader languages, GPGPU
- DirectX, OpenGL
- Vector graphics, tessellation
- Animation (skeletal, interpolation)

**DATABASES:**
- NoSQL, BASE
- Relational databases, SQL, ACID
- Distributed caching
- Data/Schema design, scalability
- Query optimization
- Data warehousing, OLAP
- ETL, Data mining
- Online data processing
- Machine learning

**CLOUD COMPUTING:**
- Cluster administration
- Xen Hypervisor
- Amazon EC2
- OpenStack

# TECHNOLOGIES

**C++:**
- C++11
- Templates, meta-programming
- STD, STL, COM / ATL, Boost
- Intel Threading Building Blocks, OpenMP
- Win API, Qt, MFC
- SSE, intrinsics, optimization
- CMake

**JAVA EE:**
- Maven
- OSGi, HK2
- JAXB, JPA, EJB, JMS
- JAX-RS, JAX-WS
- Glassfish, Jetty, Jersey, Grizzly

**C#:**
- LINQ, TPL, Async/Await
- IL, Code generation (emit)
- Native code interop (PInvoke, SWIG)
- ADO.Net
- WPF, Win-Forms
- Web services
- Ninject, xUnit, NSubstitute, NCrunch

**PYTHON:**
- Twisted, eventlet, gevent, stackless
- Python C API, native extensions
- PyQT / PySide
- IPython
- NumPy, SciPy, Matplotlib

# TOOLS

**ENVIRONMENT:**
- Windows, Mac OS X
- Linux (Ubuntu, SUSE, CentOS)
- Xen Hypervisor

**APPLICATION:**
- Mono
- EJabberd, Apache
- MySQL
- Memcached
- Riak

**LIBRARIES:**
- ZeroMQ
- DirectX, OpenGL, GPGPU (CUDA)
- NVIDIA PhysX
- LLVM, ANTLR, FLEX / BISON family

**DEVELOPMENT AND WORKFLOW:**
- Perforce, git, TFS, SVN
- MS Visual Studio, Eclipse, CodeBlocks
- IntelliJ IDEA, PyCharm
- JIRA, Confluence, Track, Bamboo, Doxygen

**DEBUGGING / PROFILING / MONITORING:**
- GDB
- Intel V-Tune, Thread Checker, Thread Profiler
- Microsoft PIX, NVIDIA PerfHUD
- NVIDIA FX Composer, ATI Render Monkey
- OpenTSDB, Graphite, Statsd, Cacti

# PROFESSIONAL EXPERIENCE

| | |
|---|---|
| Aug 2010 – Present (3.8 years)<br><br>Crytek Kiev<br>http://crytek.com<br><br>Senior Server Programmer | **Warface** (http://crytek.com/category/news/warface)<br>Award-wining free-to-play MMO FPS developed by Crytek Kiev.<br><br>Currently released in:<br>• Russia and CIS countries (Mail.Ru / commercial open beta)<br>• China (Tencent / commercial open beta)<br>• South Korea (Nexon / open beta)<br>• North America and Europe (Trion / closed beta)<br>• Brazil (Level Up! / closed beta)<br>• Upcoming releases in Turkey and Japan<br><br>In Russia game went up to 40K PCU and over 1M registered users in first month of open beta, and later set new Guinness record of 145K concurrent users playing on single realm / instance.<br><br>Warface is a first MMO project of Crytek. Server side was designed and implemented from scratch by a very small team in Kiev.<br><br>Contribution:<br>• Message bus integration and extension (Erlang, ejabberd)<br>• Database design, partitioning, sharding (NoSQL style MySQL)<br>• Lobby server architecture (C#)<br>• Server-side game features: presence, chat, game rooms, friends, equipment etc.<br>• Clustering, load balancing, high availability<br>• Multi-layer QoS system<br>• Server security: authorization system, certificates, secure protocol design<br>• Automated stress-testing and functional testing solution (Python)<br>• Operations tool: build deployment, package management, automatic updates, service monitoring, notifications etc. (Python)<br>• Engine networking library support<br>• NAT solution, Host migration support<br>• Telemetry data mining / ETL (SQL, Python, SciPy, Excel binding)<br>• Real-time monitoring system (C++, Python, Graphite, OpenTSDB)<br>• Technical and deployment documentation<br>• Managing codebases for multiple different regions / SKUs<br>• Integration with publishers' services: authentication, anti-cheat, log collectors, etc.<br>• Architecting server features and code review |
| Oct 2009 - Aug 2010<br>Crytek Kiev<br><br>Engine programmer | **CryEngine 3** (http://www.crytek.com/cryengine/cryengine3/overview)<br>Award winning multi-platform video game engine that powers Crytek's AAA game titles and licensed by many game studios.<br><br>Contribution:<br>• Linux version maintenance<br>• Automated Linux builds and development environments maintenance<br>• Engine and server side telemetry system implementation (C++/C#)<br>• Streamlined data processing DSLs implementation<br>• Telemetry warehouse architecture and ETL (MySQL + OLAP)<br>• OLAP hypervisor, data aggregation<br>• Telemetry front-ends: visualizers, analysis tools<br>• Telemetry integration with Sandbox editor<br>• Statistical telemetry analysis backend (heat maps, histograms etc.) |
| Sept 2007 - Mar 2009<br>**INTSPEI** (1.5 years)<br>Senior Programmer | **Balance** – Electronic version of accountant journal. Database client with rich editing functionality, supported database updates via e-mail.<br><br>Contribution:<br>Program architecture, database encryption using hardware security keys, data access layer, update generation logic (C#, MySQL). |

**Balance II** – Second version of Balance program, rewritten using asynchronous database interaction pattern, supports online updates of database and binary files, uses DevExpress library for GUI.

Contribution:
Program architecture, multithreading, core and database layers, update generation logic, and update sending service (C#, My SQL, DevExpress).

**P-Navigator** – Innovative desktop application for professional Internet search.

Contribution:
Architecture and code refactoring, tag and server cloud plug-ins, semantic map plug-in (WPF, asynchronous core, plug-in oriented architecture).

---

2006 – 2009
D&T research lab

**Research projects in parallel programming**. Multi-threading electronic circuits simulation engine. Parallel Boolean function minimization library.

**Boundary scan wrapper generation tool** – given electric circuit design in HDL tool generates a wrapper for top-level module compatible with IEEE Boundary Scan 1500 standard; hybrid C++/Python system; PyQt4 user interface. Contribution: system architecture, core, VHDL lexer and parser (using FLEX and BISON), VHDL semantic model; project graph, extendible design.

**Automated system for face detection chip verification** – test system that generates test input files for face detection chip based on Haar features. Generates verification scripts, and processes results of simulation. Integration with Verilog simulation environment using PLI modules. Contribution: input preparation scripts, PLI modules (Python, C++).

---

1988 – 2***
Own projects

**Several 2D and 3D puzzle games**. DirectX, DirectDraw, Direct3D. Experimenting with NPR rendering.

**Introspection and script binding generation library**. Non-intrusive reflection (introspection) library for C++. Reflection is used to dynamically create C++-Python two-way bindings. Especially suitable for applications that extend and embed scripts at the same time, like game engines.

**Extensibility platform for C++ projects** (https://github.com/mikhtonyuk/SekaiFramework)
Provides multiple solutions for developing modular extensible applications like binary encapsulation, interfaces, cross-module RTTI and class discovery. Inspired by Eclipse extensions and OSGi architectures.

**Engine for 2D and 3D visualization** (https://github.com/mikhtonyuk/3DEngine)
Highly decoupled extendable plug-in-based architecture; multithreaded design; asynchronous resource system with archive support; C++-Python script two-way bindings using own reflection library; advanced shader-based material system (uber-shaders); dynamic shader linking; post-processing support; vector math library optimized for SSE and SPU instructions; scene graph; basic animation; component-based entity system; integration with NVIDIA PhysX and NVIDIA PerfHUD.

**Hardware-accelerated scalable vector graphics UI library**. Transforms vector graphics primitives to meshes; allows creating 3D user interfaces; compatible with SVG format; oriented on in-game GUI; highly optimized (tessellation, batching, fast rendering algorithm of concave polygons); adjustable level of detail; TrueType font tessellation for high-quality scalable text.