Cannibal Game Studios
Cannibal Game Studios specializes in providing custom gaming solutions to seasoned game studios and game development novices alike. The technical realization is an important part of any game project and requires experience to be done successfully. Cannibal Game Studios’ technology makes it feasible to successfully outsource this technical realization reducing risk, costs and time-to-market.

Cannibal Game Studios aims to provide custom tailored solutions which can be rapidly developed and maintained in order to match the needs from an industry with ever-changing technology and requirements.

Customization
Next to deploying technology for interactive 3d application and games, a major focus is customized game middleware for the creation of virtual worlds and video games. Game developers often make use of expensive, bulky, suites of generic game development tools which contain many unnecessary features on one hand and need to be heavily customized for a particular end-product on the other. An alternative would be to develop such tools completely in-house. Either way, a lot of effort is spent outside the core-business of the game developers; instead of focusing on making good games, they need to focus on making the tools to make games in the first place.

Using extendible frameworks components and plugins we can easily select the functionality we want and provide customer specific functionality, leading to a product tailored to the customers unique need. Our main technology consists of a large number of components and plugins for the following three main frameworks:

1. Cannibal Framework: our toolbox for programming;
2. Cannibal Engine: our 2d and 3d gaming and rendering engine;
Projects
Improving upon our existing technology enables us to develop custom solutions faster, with more reliability and with lower cost, which in turn allows us to better meet the needs of our customers. We are looking for talented students who would like to join us and work on one of the following research topics in the area of software evolution:

Software Configuration Management
Customers generally have very specific requirements for their gaming products. In practice, this means that the reusable components and the frameworks that make up the software need to be modified. Many (sometimes conflicting) changes will have to be made to facilitate our different solutions. Since we use an agile approach to software development, changes occur often and are certain.

How do we effectively and efficiently manage and track all these changes to our frameworks and components and propagate them to the correct products and versions of products we release and have released to customers?

Backwards Compatibility
Users of our custom middleware solutions use that middleware to create their own artifacts. When a new version of the middleware is released, there exists a possibility that compatibility with current artifacts will be broken. Straightforward examples include changing the API, which breaks the source code, but also changing the abstract data model we use to represent creative artifacts (e.g. levels), which could potentially render the stored artifacts useless. These changes are mostly of syntactical nature, but changes in the semantics of certain properties or methods may also be introduced (e.g. changing the meaning of a certain render setting), often impacting backward compatibility in less obvious ways.

How do we ensure that we cause no harm to the end-users artifacts when we make changes to our components and framework? In other words: how can we (automatically) transform the end-user artifacts so that he experiences no errors (no syntactical incompatibility) and have them retains its meaning (no semantic incompatibility)?

Model Driven Development
Currently configuring the different compositions of plugins and settings is mostly done “by hand” in our code. For this we have a lot of glue code that specifies how particular components need to be stitched together into a final application. This has as effect that the level of customization we can provide using this approach, and the speed at which we can create new solutions, is lower than could be theoretically be achieved. Since writing glue code by hand allows room for errors, more maintenance than necessary has to occur.

What model driven engineering approaches can we use to provide us with a successful custom gaming solution software model to both speed up our development and reduce maintenance effort? Can we develop a model driven development environment tailored towards specifying these custom gaming solutions?
Internships at Cannibal

Cannibal Game Studios is looking for talented and pro-active students that have a passion for good software engineering. A passion for games and know-how on game development is definitely a plus.

The internship will be performed in a research team, where we attempt to cover the three subjects above and work on further improving our software architecture all at the same time. Because of the nature of the internships, you will be expected to perform the work that forms the necessity for the research described above: building custom gaming solutions for our customers one or two days a week. This experience will allow you to more concretely grasp the problems and will serve as a testing ground for your theories and proposed solutions.

Do expect us to start using your work and recommendations and expose them to the trials and tribulations of daily life during your internship!

Contact
For more information or application for one of the positions/research topics, please contact:

Jeroen Dobbe
COO

+31 (0)15 711 2718
j.dobbe@cannibalgamestudios.com