UNITY TEST

Congratulations, you have made it to the next phase of DTT's application procedure!

This phase will consist of a development test made in the Unity Game Engine. During your time at DTT, you will work for strong international brands, including Greenpeace, Randstad, ING, Philips, Rabobank, FOX Sports, and KPMG. Through this test, you can demonstrate that you are capable of contributing to DTT and her clients. DTT promises her clients a detail orientated approach, a high quality standard for deliveries, and a work ethic that goes that extra mile: we look for applicants that understand these values.

This test gives us a better understanding of how you work, and provides you with insight into what you can expect when working at DTT.

Keep in mind:

I. Log the hours you spend on the project in the provided Excel document, provide a detailed description of your activities to help us in the grading process.
II. We strongly advise you to complete the **user stories** within 20 hours, and to adjust your scope based on this advisory deadline. Time taken for **bonus points** will not be included in the final time evaluation.

THE TEST

We challenge you to procedurally generate a perfect maze using an algorithm of your choice. A perfect maze is defined as having no loops or closed circuits, and no inaccessible areas. In a perfect maze, any two points must be connected through a singular, unbroken path. There are multiple algorithms to generate such a maze, as is detailed on this Wikipedia page.

User stories

DTT uses user stories as a baseline for all our (technical/design) documentation. Make sure your test adheres to the user stories and conditions of satisfaction as stated below.

For the minimal viable product, the following user stories should be implemented:

User story 1: As a developer, I want to generate a perfect maze, so that I can showcase my technical capabilities.

- *I want to implement a perfect maze generation algorithm.*
- *I want the algorithm implementation to create a visual representation.*

User story 2: As a user, I want to be able to configure the maze, so that I can control its visual representation.

- *I want to be able to set the width of the maze in the user interface.*
- *I want to be able to set the height of the maze in the user interface.*
- *I want to be able to generate an unevenly sized maze.*
- *I want to be able to start generation from within the user interface.*
- *I want to be able to (re)generate the maze at any time.*
- *I want to be able to see the maze in its entirety.*

User story 3: As a user, I want to generate mazes of varying sizes, so I have a better view of the algorithm's visual representation.

- *I want to be able to configure the maze's size to a maximum of 250x250 cells.*
- *I want to be able to configure the maze's width and height to at least 10x10 in size.*
- *I want to be able to generate a maze without significant impact on performance.*

User story 4: As a user, I want a responsive user interface, so that I can view the maze on multiple device resolutions.

- *I want the user interface to look good on Desktop (1920x1080).*
- *I want the user interface to look good on iPad (2048x1536).*
- *I want the user interface to look good on iPhoneX (2436x1125).*

Additional requirements:

I. Use a Unity version of 2020.1 (or newer).
II. Your project should compile in the Unity editor without errors.
III. Do not use the Unity Asset Store for anything directly related to the essence of this test. Assets related to design, animations, particle effects, etc. are allowed.
IV. When consulting an online source for the generation, please refer to this in your hour logs. We are fine with a tutorial being used as long as the test deviates/adds enough for us to grade your ability to create something of your own.
V. Use C#.

Bonus points for a good impression (all optional)

As a Unity intern, you are expected to be productive across the full stack of game development, from writing back-end code to crafting the user experience, UI-animations and gameplay. We are more likely to invite candidates for a follow-up interview when we can accurately gauge one or more of these aspects within game development. As such, the bonus points are very 'free format', allowing you to decide which impression you want to leave us with.

A few ideas for extra points:

I. Add new features (eg. switching between different generation algorithms).
II. Get creative with animations (eg. step-by-step generation animations).
III. Go all-in on visuals (eg. with shaders, 3D models, particle effects, etc).
IV. Out-of-the-box cell and maze shapes (eg. hexagonal cells or a round/organic maze).
V. Go big with very (very) large mazes (eg. through compute shaders).
VI. Thoroughly document your code/architecture (eg. with a class diagram).
VII. Have a fun idea? We'd love to see it :)

These points are completely optional and do not count towards your 'total time'. We do not negatively judge a test that does not include bonus features, but differentiating your test does increase your chances to get invited for a follow-up.

Delivery

When you are satisfied with your game, **zip** your project (including your hour log and documentation) and send a WeTransfer link to your person of contact at DTT.

If you worked with a GIT repository, please follow the steps below:

I. Invite *apply@d-tt.nl* as a contributor with Admin privileges
II. Make sure that the repository includes the full project source code
III. Make sure that the repository includes a copy of any other deliverables e.g.:
    A. your completed hours log
    B. your class diagram
    C. etc.

Our review

We review your test on different criteria. Functional requirements are graded on their completeness and the chosen method of implementation. The quality and structure of code are graded on (among other factors) consistency, clarity, optimisation, extensibility, reusability, and single-responsibility. The hour log is graded on its completeness, language, and level of detail. Your code will undergo the most scrutiny during our review, make sure it is well-structured and extensively commented.

Among other factors, the following points are essential to us:

I. **Consistency is key**: naming, structure, approach. We do not judge a chosen convention: we do judge if the chosen convention is adhered to consistently.
II. **Clarity**: comments are used effectively (and consistently) to make the code clear and understandable.
III. **Single responsibility**: the code is maintainable, isolated, generalised, reusable. Always avoid duplicate code.

Adherence to these criteria indicates an approach to work we greatly value.

Questions?

If you have any question regarding the requirements of the assignment, or if you are stuck on a problem for too long, please don't hesitate to contact us.

More DTT

Feel free to have a look at all our apps at: https://www.d-tt.nl/en/apps.