RUTGERS

Rutgers - Computer Graphics Course - Fall 2015 - Assignment B1

Unity: Navigation & Animation

This is a group assignment.

PART0: Setup Git Repo for Unity

First, you must set up a *NEW* git repo on which your whole team can collaborate. The purpose of having a git repo is mostly for you to be able to work on your projects together, without necessarily having to meet in person. Using git, specially when working on a group, will make creating a project not only easier, but also much more efficient. So get start using your git repos, and feel the magic of git! (Refer to "Git Introductory Tutorial" available on sakai resources to learn about git.)

NOTE:

1. This git repository is independent of the one you are using for the SteerLite (A) assignments.
2. You will create one git repository to be used for all Unity (B) assignments. Each group will thus maintain 2 git repositories, for SteerLite (A) and Unity (B) assignments respectively.

Follow these instructions and setup your group git repo for Unity projects:

1. From the link below, "fork" the template structure of unity projects into your group account on GitHub (refer to A1 instruction if you have not created a group account yet): https://github.com/mahyarkoy/UnityProjects
2. Now each member of your group can clone the repo and start using it. Read here for setting up line ending in your specific platform (win, linux, mac) so members on different platforms can work together: https://help.github.com/articles/dealing-with-line-endings/
3. Change README.md file to your personal information. Add your team logo and website (see part 4 for team website creation info). Update README.md as you develope each

unity project. Of course remember to push to repo after you make changes, and always to pull before you start changing/adding new things.

4. Make sure not to change the implemented structure of file system.
5. Look inside .gitignore text file: this is where you can specify what files should be ignored by git when it pushes/pulls/clones. Do not remove/change anything on the file. Read about .gitignore and its syntax here: https://git-scm.com/docs/gitignore

PART 1: Navigation [5 points]

In this part you must create a simple crowd simulator (navigation system), where you can select a set of agents (using a mouse interface) and can click on a desired location in the environment to have them navigate towards.

Useful materials and readings:

1. Unity Navigation Tutorials: http://unity3d.com/learn/tutorials/topics/navigation
2. Unity Navigation documentation: http://docs.unity3d.com/Manual/Navigation.html

Follow these instructions and create your simulator:

1. Design a relatively complex environment using different obstacle configurations. Your environment must include a simple maze field, several rooms, bottleneck areas, at least 3 height levels (e.g. a couple of containers on top of which agents can move/jump is considered a second level), and connections between levels (stairs, bridge, etc.).
2. Compute a navigation mesh for the whole environment (2D planes on different levels).
3. Create an "Agent" prefab with NavMeshAgent component to allow agents to navigate in the environment while using the navigation mesh.
    a. The Agent can be a simple capsule with a capsule collider for now
    b. You can instantiate multiple instances of the Agent prefab to create a crowd of agents in your scene
4. Create a simple mouse script to select agents and specify which destination to navigate towards. Read about camera rays to understand how to detect a mouse click position in your scene: http://docs.unity3d.com/Manual/CameraRays.html

5. Setup a "Director" script that keeps track of all the selected characters and sends messages to each character's component to move to the specified destination.
6. Use NavMeshObstacle to create obstacles in the environment which can be moved around. Obstacles should be selectable, and movable using arrow keys. The obstacles must carve the navigation mesh (check the carve checkbox in NavMeshObstacle).
7. Use off mesh links to connect disconnected navigation meshes to implement jump between different planes on different heights (hint: off mesh links can only be created between two surfaces, that means you may have to define a null object on the surface of another object).
8. Explain the difference (in behavior) between carving and not carving option for a NavMeshObstacle? When and why should the carve checkbox be active/deactive? Describe the problem with these two situations:
    a. if we make all obstacles carving.
    b. if we make all obstacles not carving.
9. Describe a way for implementing how an agent can avoid obstacles with not-carving option? (hint: a way was discussed in details in class).
10. **Extra credit:** Create several obstacles that automatically move in your environment (e.g. some roaming devils!), and make it such that your crowd avoids them! If you make them carve the navigation mesh, would there be any problem with your crowd movement in this case? What is the problem? What is the reason?
11. **Extra credit:** Add some “Nazgûl” agents! These agents can be selected and moved like others, but the other normal human agents should always avoid them at any cost, by a rather far distance! So for example if one of them enters a room, all human agents in that room should run away to other places, and when human agents are navigating, they should avoid these agents! Remember: Nazgul agents must be able to move seamlessly, like any other agent. Mention in your documentation how you implemented this part. (hint: it is possible to use both NavMeshAgent and NavMeshObstacle on same agent, but it is tricky to handle that, very tricky! some good/smart scripting is needed.)

PART 1 Deliverable: A web playable demo of an interactive crowd simulator where the player can select one or more agents and issue commands of where they must navigate by clicking on a location in the environment.

PART 2: Animation [5 points]

In this part you should use Mecanim Animation System to create some animated agents. These agents should be able to walk, run, and jump (among other exciting thing you can make them do).

Useful materials and readings:

1. Mecanim documentation: http://docs.unity3d.com/Manual/AnimationOverview.html
2. Mecanim example projects on asset store:
    a. https://www.assetstore.unity3d.com/#/content/5328
    b. https://www.assetstore.unity3d.com/#/content/7673

Characters:

1. Male : https://www.assetstore.unity3d.com/en/#!/content/124
2. Female : https://www.assetstore.unity3d.com/en/#!/content/127
3. Robot : https://www.assetstore.unity3d.com/en/#!/content/4696
4. Soldier : https://www.assetstore.unity3d.com/en/#!/content/122

Animation data for Mecanim:

https://www.assetstore.unity3d.com/#/content/5330

(look at the asset store for more characters and assets)

Follow these instructions and create your prefab of an animated agent:

1. Create an animated human character using Mecanim. Take a humanoid character, import animation data onto the humanoid (you can get the animations from the link above), and create a simple animation state machine to have him walk, run, jump.
2. Create a simple controller script where a player can control the character using arrow keys. Use run modifiers (e.g., shift key) and SPACE to jump.
3. Create a prefab of the animated character.
4. Set up a “third person” camera view which follows the character and is located behind and just above the shoulder (RPG style).

PART 2 Deliverable: A web playable demo where you can control an animated character using the keyboard. Third person camera view is required.

PART 3: Combine Animation & Navigation [5 points]

In this part you must combine part 1 and part 2 together, so that you can navigate crowds of your animated human characters with meaningful animation during each action in your crowd simulator.

Follow these instructions:

1. Integrate your prefab(created in part 2) into your crowd simulator (part 1) to create a crowd of directable animated characters.
2. Characters should jump when navigating off mesh links.
3. You should be able to set the desired speed of the character(s) and they should transition from running/walking (e.g. by multiple clicks on a target.)
4. **Extra Credit:** Think of other off mesh links you could implement and their associated behaviors. Implement several interesting offlink meshes, and create animation behaviors in your prefab for that movement (e.g. fly, crouch, swim, etc.). Add it to your simulator.

PART 3 Deliverable: A web playable demo of an interactive crowd of animated characters.

PART 4: Webpage (Blog)

Create a blog for your team online. You can use blogger to create your team blog. You may also host the website yourself if you wish.

In your blog, you must create a post for each of your projects (A and B assignments) with the following information:

- embedded links to your youtube videos.
- brief description of what you did in the assignment and what we see in the videos.
- links to web playable unity demos (see instructions below for hosting your game online).

This will eventually become a very good webpage for you to present your works.

To upload your game online: either find a host, or use facebook. For uploading your game on facebook, follow this link:

http://answers.unity3d.com/questions/519982/how-to-publish-my-game-as-a-facebook-app.html

SUBMISSION:

Submit the following in Sakai for grading:

1. Your Unity project in a zip file which contains the Assets/ folder and includes all your C# scripts.
2. Offline builds of your 3 demos (part1, part2, part3), including the html file of your project and all required run files. Remember, it is your responsibility to make sure your offline version works fine when downloaded, so check several times before submission. If your games cannot be played, you get 0 (no warning or announcement will be given this time!).
3. Brief documentation about your project. Clearly describe how your project works and your extra credit attempts. This is your guide for the user. For unmentioned/unclear aspects of your project, you won't get any mark. So please write your documentation brief, clear, and complete.
4. Link to blogger post or your web-page, where there is a post about this project, containing the documentation and 3 links to your ONLINE PLAYABLE demos.
5. **Extra credit:** Integrate the 3 scenes (all 3 parts) into one single demo with a game menu allowing to navigate between part 1, 2, and 3.

Please submit IN TIME! even if you miss the submission by seconds, you won't be given any marks. So upload well before the deadline to avoid last minute problems.

NOTE: Extra credit will be given at the discretion of the instructor.

--

Good Luck Everyone!