PLAYWARE

Bem-Vindo ao Projeto Raining Days

O projeto é um game Open-Source. Toda e qualquer pessoa pode participar contribuindo com pequenas coisas, como assets para o game, revisão de códigos, novas funcionalidades e etc.

O game tem uma estrutura bem simples, mas o objetivo é claro: criar um game-AAA com ajuda da comunidade e revisão por pares. Utilizando-se de recursos que os criadores de conteúdo julgarem melhores para si próprios.

Claro que todo e qualquer novo asset ou revisão passará pelo crivo do idealizador do projeto e também todas as versões disponíveis no projeto para download apresentarão todas as modificações aceitas. Importante ressaltar que não serão todos os assets e features que serão aprovados, porém para toda e qualquer modificação aceita, o criador receberá os devidos créditos.

O game não tem uma data prevista para conclusão e lançamento, mas o seu andamento e acompanhamento acontecerá via livestream e qualquer um pode assistir ou ajudar na idealização do game.

- Para acompanhar as lives, visite: twitch.tv/kaynao13
- Para baixar a sua versão do projeto e alterar, realize o comando **git clone** do projeto aqui apresentado: https://github.com/kaynan013/RainingDays0.1

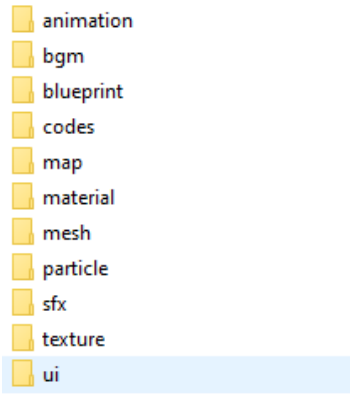
Periodicamente será liberado um canal aberto de discussão no Discord para a discussão pública do projeto, fique atento às lives para ter mais informações a respeito.

Abaixo seguem todas as diretrizes e como você pode submeter e contribuir para o projeto. Note que estas regras devem ser seguidas à risca. Não haverá qualquer punição para quem não as seguir, mas a alteração simplesmente não será aceita.

Sumário

Diretrizes Gerais

- Será utilizada a interface GIT para a organização e versionamento do projeto;
- Será utiliza a Unreal Engine 4 como plataforma de criação. Não se deve alterar a Engine com a qual o game será criado;
- Todos podem participar e contribuir com grandes e pequenas coisas. Seja um asset, uma alteração no código ou uma nova funcionalidade;
- Para aceitação da alteração desejada ou do material, o criador deve:
    - Empacotar todo o conteúdo, por exemplo: se fizer uma mesh já texturizada, deve enviar a mesh e as texturas;
    - Deixar claro em um documento de texto junto à pasta (.txt) qual a alteração desejada e todo o conteúdo da alteração descrito, siga os arquivos de exemplo em **Guidelines** caso tenha dúvidas;
    - Assets podem ser enviados como **.zip**, desde que devidamente apresentado, como indicado acima;
    - Siga o arquivo "Tipos de Arquivos.pdf" em **Guidelines/Regras** para mais informações de o que pode ser enviado ou não e o que cada arquivo pode conter;
    - No documento de texto, que deve ser submetido junto das alterações ou novo conteúdo, deve estar declarado o que o próprio POSSUI e NÃO-POSSUI conforme o modelo **"orientação_de_exemplo.txt"** disponível em **Guidelines;**
    - Não há necessidade de todos os novos conteúdos terem todas as especificações apresentadas, porém, caso falte algum dos itens do campo "Pode conter" de cada asset, deve informar no arquivo de texto o que **falta;**
    - Algumas extensões de arquivo têm uma especificação, o criador deve seguir o que está apresentado na seção **Diretrizes Específicas** para cada extensão;
- Todas os novos assets devem estar devidamente organizados na pasta do projeto em **RainingDays\Content\Comunidade.** Devem estar devidamente separados a depender do tipo de asset que será enviado;



- O criador deve organizar seu conteúdo conforme a estrutura de pastas sugere;
- Todas as alterações devem ser submetidas via pull-request pelo GitHub;
    - Toda pull-request deve informar na mensagem qual a alteração realizada ou qual novo conteúdo inserido;
- Caso o criador sinta a necessidade de criar uma nova pasta, esta alteração deve estar descrita no arquivo de texto ou no campo de texto da pull-request;
- A princípio, o game é pensado para um formato 1920px x 1080px. Toda nova funcionalidade ou asset deve ser criado para este formato;
- O nome do arquivo a ser submetido não deve conter o mesmo nome de outro arquivo, qualquer alteração deve ter um nome de arquivo único. É recomendado pesquisar no projeto clonado o nome do arquivo antes de o criador salvar o seu próprio. Caso o criador desejar, pode versionar seu arquivo paralelamente a outro, exemplo: aldeaocomum_0, aldeaocomum_1, aldeaocomum_2 e assim por diante;
- Caso prefira utilizar um software que não os utilizados pelo idealizador do projeto, o arquivo a ser submetido deve obedecer às regras dos tipos de arquivo apresentadas em **"Tipos de Arquivos.pdf"**;
- Não garantimos a integridade da autoria dos conteúdos, mas atribuímos completamente caso seja inserido ao game, então recomendamos fortemente que tenha o seu asset ou código apresentados em uma interface externa, por exemplo: o código salvo em algum repositório no GitHub ou o seu modelo 3D apresentado no Behance ou ArtStation. A autoria será atribuída para quem:
    1. Submeter a alteração com a autoria devidamente reivindicada no arquivo de texto (.txt) que deve ser enviado e;
    2. Caso haja necessidade de reivindicar a autoria por um conteúdo que tivera erroneamente a autoria atribuída, será considerado, nestas plataformas externas, a autoria do conteúdo apresentado com a data mais antiga;

A produção

A produção do game está dividida em quatro fases, algumas etapas e para cada etapa, algumas sub-etapas. Cada fase de produção estará devidamente indicada nos números de versionamento do projeto. Perceba que este repositório tem, à frente de seu nome, um número de versão. Este número de versionamento mudará a cada nova versão estável do projeto após, claro, alguns testes. Perceba, também, que este documento também tem um número de versão. Este documento será atualizado a cada fase concluída. Há um código de cores na coluna "Duração" que não representa muita coisa, apenas uma previsão de quanto tempo cada sub-etapa levará.

Fase	Etapa	Sub-etapa	Duração
Concept	1	Softwares, TI-Infra e termos	
		Briefing e Brainstorming	
		Roteiro (pontos principais)	
		Roteiro (Cenas)	
		Roteiro (Sidequests)	
		Mecânica	
		Sistema	
		GDD	
	2	Id. Visual e Moodboard	
		Mundo	
		Personagens	
		Assets	
		Storyboard	
		UI	
		Jogabilidade	

A fase atualmente é a de Concept, onde não existe qualquer produção efetiva de material para o game, somente algumas diretrizes para a fase de produção.

Qualquer pessoa pode testar o game e tecer comentários a respeito, também está livre para hackear o game ou procurar por bugs. O arquivo **Cronograma,** em **Guidelines/Regras,** será constantemente atualizado conforme novas sub-etapas forem concluídas.

Vale ressaltar que a produção não é localizada, junto do idealizador do projeto, qualquer pessoa pode contribuir para o enriquecimento da mesma. Visto que muitas cabeças pensam melhor do que uma.

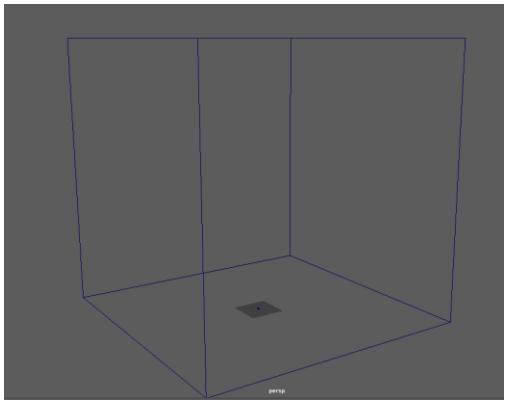
Diretrizes Específicas

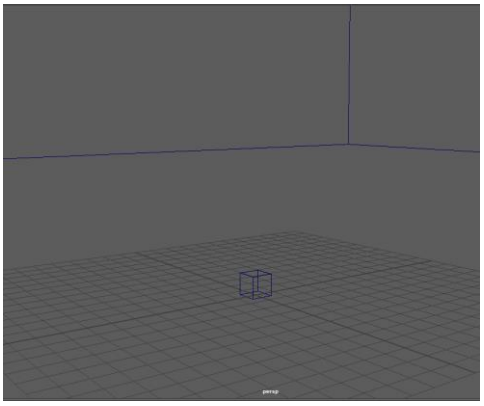
Esta seção é dedicada a formular como que cada tipo de conteúdo pode ser enviado, algumas regras precisam seguir a orientação aqui formulada, outras ficam a critério do criador do conteúdo.

3D

Extensões de arquivo aceitas: .obj, .fbx

Todo objeto 3D pode ser separado nos três tipos de conteúdo abaixo formulados e devem seguir as orientações especificadas. Em **Guidelines** há o arquivo “proporcao.fbx” que deve ser utilizado como uma especificação do tamanho de objeto a ser enviado. Há dois cubos no arquivo:





O maior (à esquerda) especifica a altura total do manequim de exemplo (“ManequimDeExemplo.fbx”) que a Unreal exporta, ou seja, pode ser utilizado como comparador de proporção para qualquer modelo a ser exportado. O criador pode utilizá-lo para redimensionar sua própria mesh e exportar com os tamanhos corretos qualquer modelo. O cubo menor (à direita) mostra o tamanho de um objeto 1:1 no Maya. Todo e qualquer objeto pode ser feito utilizando o cubo menor como unidade de medida e aumentado ao tamanho do cubo maior na exportação do modelo.

Arquivos de meshes podem ser exportados nas extensões .fbx e .obj, somente. Para se enviar um arquivo na extensão .obj é recomendado que não seja enviado junto o .mtl. Todo material e textura pode ser enviado junto, mas

texturizado separadamente e cada textura enviada separada do arquivo 3D junto no pacote.

Mesh

Cada objeto não necessariamente precisa ter um arquivo separado. Um mesmo arquivo (.fbx) pode conter mais de um modelo, desde que devidamente sinalizado no arquivo de texto (.txt) que deve ser enviado junto.

A malha pode ser triangulizada ou quadrangulizada, malhas que tenham faces não triangulares ou quadrangulares podem ou ser solicitadas para alteração ou não aceitas. Vale ressaltar que não necessariamente devam ter as proporções acima indicadas, mas se tiverem não terão alteração alguma por parte do detentor do projeto do game.

Todas devem ser mapeadas. Não serão aceitas malhas não mapeadas e, malhas mapeadas que não estejam de acordo com os padrões estabelecidos abaixo serão solicitadas de alteração.

Não há necessidade de a malha ser texturizada. Caso a malha seja texturizada, o material deve ser devidamente sinalizado e cada canal devidamente indicado. Nesta situação, todas as texturas devem ser enviadas juntas no pacote e o arquivo de texto (.txt) deve conter as informações de extensão das texturas e qualquer outro parâmetro adicional.

Toda mesh deve ser otimizada (reduzida), tendo seu poly-count não superior a 30.000 faces (para assets, para personagens este número é discutível). Note que, a depender do tamanho do objeto, não há necessidade de um poly-count muito bem definido e será solicitado ao criador reduzir ou aumentar o número de faces a depender da demanda ou da vontade do idealizador.

Uma mesh pode ter mais de um canal UV, desde que devidamente sinalizado.

Rig

Os rigs podem ser enviados separados das malhas, contudo, se este for o caso, cada joint do rig deve ser devidamente indicada e parenteada, conforme o modelo “ManequimDeExemplo.fbx”. Note que o modelo de exemplo agrupa todo o rig e o tranca para não haver edição dos bones, também esconde este grupo e trava edições indesejadas (no exemplo, as transformações de scale estão indisponíveis).

Todo rig deve ter controladores (conforme o arquivo de exemplo).

Animação

Toda animação deve obedecer os bones. Nenhuma animação deve adicionar ou remover bones. Caso uma transformação seja impossível sem adicionar ou remover bones, um novo rig deve ser criado. Toda animação deve corresponder a um grupo de bones já existentes dentro do projeto. Caso o criador prefira enviar um modelo completamente próprio, o arquivo deve conter todos os itens acima e também as texturas correspondentes.

A animação pode não conter a malha (somente bones animados), para isto, cada animação deve estar devidamente sinalizada e, no arquivo de texto que deve ser enviado junto (.txt) indicar à qual malha pertence aquela animação.

Textura

Extensões de arquivo aceitas: .png, .jpg, .tga, .psd

Toda textura deve, no mínimo, exercer algum dos papeis abaixo:

- Pintar um objeto (albedo/cor-base/diffuse);
- Trabalhar a iluminação de um objeto (specular/metálico/roughness);
- Sombrear um objeto (normal map/ambient occlusion);
- Não serão todos os objetos que terão displacement map, somente personagens e assets grandes (mais da metade das proporções do cubo apresentado no arquivo **proporção.fbx** em **Guidelines**, ou a ser indicados pelo idealizador do projeto;

Portanto não serão aceitas texturas que não texturizem um objeto já existente (texturas "soltas"). Todos os objetos disponíveis podem ser texturizados ou retrabalhados a depender da vontade do criador.

Toda textura deve ser submetida como imagem em uma das extensões abaixo mencionadas, num tamanho máximo de 4096px. Toda textura deve ter, ao fim do nome do arquivo, uma indicação de:

1. À qual mesh aquela textura se refere;
2. Qual canal ela preenche. Indicado no nome do arquivo da seguinte forma:

a. "_alb" para cor-base/albedo/diffuse;

b. "_spec" para specular;

c. "_met" para metálico;

d. "_rough" para roughness/aspereza;

e. "_norm" para normal map;

f. "_disp" para displacement;

g. "_aocc" para ambient occlusion;

h. Caso haja máscaras, deve estar indicado "_maskAocc", ou "_maskAlb", ou "_maskSpec" e assim por diante;

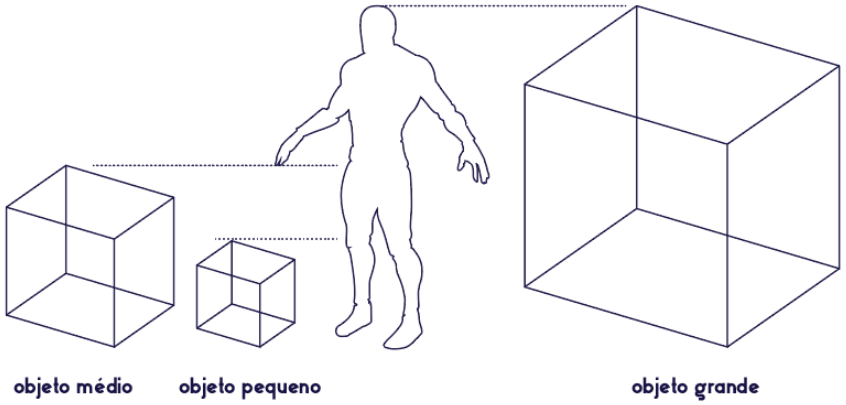
3. Deve seguir o exemplo: “objeto_canal.ext”;
4. No arquivo de texto que deve ser enviado junto (.txt) devem constar todas as texturas criadas;

Texturas que estejam incompletas serão descartadas. Texturas que não estejam com o nome conforme padronizado acima estão sujeitas à solicitação de alteração ou serão descartadas.

As texturas podem ser enviadas como um asset da engine (.uasset). Nesta situação, o criador deve indicar no arquivo de texto (.txt) em qual versão da engine o asset foi criado.

As texturas devem obedecer à proporção do objeto desejado. Abaixo seguem alguns padrões que podem ser seguidos. Note que são um exemplo, não significa que devam ser desta forma sempre. Para assets 3D, toda textura deve estar dentro de uma região quadrada, não serão aceitas texturas que tiverem uma área retangular. A preferência pela extensão de arquivo a ser utilizado é a .tga, contudo, outras extensões podem ser utilizados. Não serão aceitos arquivos .tga de 32-bits.

A figura abaixo exemplifica tamanhos que podem ser utilizados para as texturas. Note que a depender do tamanho do objeto que será texturizado, haverá a necessidade de uma textura maior (como os considerados “objetos grandes”), neste caso, toda textura que precise ser maior do que o tamanho máximo indicado no início desta seção, deve manter o tamanho máximo e a mesh que deverá ser alterada, sendo criada para ela um novo canal UV.



Tamanho do objeto	Opção 1	Opção 2	Opção 3

Objetos grandes	4096px	4000px	3200px
Objetos médios	4096px	3000px	2048px
Objetos pequenos	2000px	1800px	1024px

JPG

Esta extensão será utilizada como a mais adequada para assets e texturas que não precisem demandar tanto processamento (para objetos simples, em que o jogador não precisará de tanta interação, como pedras, texturas de background e itens que serão apanhados pelo jogador).

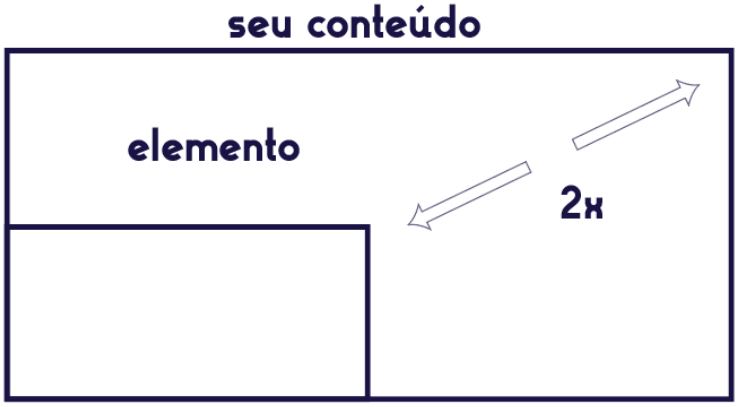
A extensão JPG, a princípio, não comprime transparências, contudo há formas de fazê-lo a depender da forma da compressão. Toda e qualquer compressão do arquivo que não o formato padrão .jpg será prontamente descartada. Para a utilização de transparências, deve-se utilizar as extensões abaixo.

PNG

Esta extensão será utilizada, a princípio, para HUDs, UI do game e partículas. Não será utilizada para texturizar meshes. Qualquer objeto texturizado com esta extensão terá as suas texturas descartadas.

Todas as texturas desta extensão serão aplicadas em elementos 2D e, conforme o item anterior, devem desempenhar uma função para algum elemento já existente. O criador pode consultar elementos já existentes em “RainingDays\Content\Comunidade\ui”.

Toda textura ou elemento gráfico deve obedecer à proporção do elemento que a receberá, evitando “esticar” o elemento. Todo novo asset pode, no máximo, estar duas vezes maior que o elemento.



PSD

Os arquivos de Photoshop são muito pesados, portanto serão utilizados em casos específicos, para texturizar elementos chave do game. Caso o objeto texturizado com esta extensão não demande o tipo da extensão enviada, a textura pode ser descartada ou solicitada de alteração ou salvamento em outra extensão.

TGA

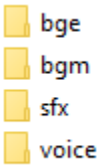
A única restrição para esta extensão de arquivo é que não deve ter compressão de 32-bits.

Áudio

Extensões de arquivo aceitas: .mp3, .wav

Não há muitas restrições aqui. A não ser pelo fato de que as duas únicas extensões aceitas devem deter a especificação contida em **Tipos de arquivos.pdf.**

Todo novo arquivo de áudio, ou qualquer arquivo de áudio modificado, deve estar devidamente organizado nas pastas. Da seguinte forma:



- BGM: para o áudio específico de músicas de fundo. Qualquer áudio que não for inserido diretamente no game e provocado por algum asset ou personagem, pode ser categorizado como bgm;
- SFX: para o áudio que é provocado por uma ação ou evento no game, como: passos, zunido de insetos, disparo de arma, etc.;
- BGE: para qualquer áudio que possa ser inserido no background do game, mas não provocado por um evento, exemplo: farfalhar de árvores, pingos de chuva, uivar de lobos à distância, etc;
- Voices: dublagem.

Texto

A princípio o game deve ter pouco texto, porém existem duas camadas de interação com o jogador que devem ser devidamente esclarecidas:

Desenvolvedores x jogador;

Game x jogador;

A camada de interação que solicita ações externas ao jogo é a dos desenvolvedores, é o tipo de interação burocrática, como atualizar informações ou completar cadastro em algum lugar. Esta deve utilizar os princípios da

identidade visual da empresa, disponível em “Manual ID. Visual Playware.pdf”, em **Guidelines.**

No que tange à identidade visual do game e como o game irá interagir com o jogador é onde o material textual será produzido. Os criadores pode criar a tipografia do game, a tipografia deve ser exportada devidamente no formato TrueType.