Find the project web page here:
http://yaksoy.github.io/parallaxbg/

Parallax Background Texture Generation

Brigham Okano*
Simon Fraser University

Shao Yu Shen*
Simon Fraser University

Sebastian Dille
Simon Fraser University

Yağız Aksoy
Simon Fraser University

Figure 1: We present a framework to turn a landscape photograph (left) into a layered game texture (right) with parallax effect.

ACM Reference Format:
Brigham Okano, Shao Yu Shen, Sebastian Dille, and Yağız Aksoy. 2022. Parallax Background Texture Generation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters (SIGGRAPH '22 Posters), August 07-11, 2022*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3532836.XXXXXX

1 INTRODUCTION

Art assets for games can be time intensive to produce. Whether it is a full 3D world, or simpler 2D background, creating good looking assets takes time and skills that are not always readily available. Time can be saved by using repeating assets, but visible repetition hurts immersion. Procedural generation techniques can help make repetition less uniform, but do not remove it entirely. Both approaches leave noticeable levels of repetition in the image, and require significant time and skill investments to produce. Video game developers in hobby, game jam, or early prototyping situations may not have access to the required time and skill. We propose a framework to produce layered 2D backgrounds without the need for significant artist time or skill. In our pipeline, the user provides segmented photographic input, instead of creating traditional art, and receives game-ready assets. By utilizing photographs as input, we can achieve both a high level of realism for the resulting background texture as well as a shift from manual work away towards computational run-time which frees up developers for other work.

*Denotes equal contribution.

SIGGRAPH '22 Posters, August 07-11, 2022, Vancouver, BC, Canada

ACM ISBN 978-1-4503-9361-4/22/08…$15.00
https://doi.org/10.1145/3532836.XXXXXX

2 PIPELINE

Our proposed framework consists of four building blocks. We show an overview of the pipeline in Figure 2:

Depth-based interactive segmentation. As a first step, we segment the input photograph into depth layers to later apply the parallax effects. In our examples, we generate depth estimates using [Miangoleh et al. 2021]. Both the MiDaS [Ranftl et al. 2020] and LeRes [Yin et al. 2021] version provide useful information that complement each other during segmentation. We decide to choose the best of both for each layer individually. We then apply a soft threshold on the depth map to define the desired depth range for each layer. The adjusted depth map can then be used directly as a segmentation mask for the input image. Alternatively, segmented images from other sources can be used or segments from different images can be mixed. To ease the next processes and increase the visual quality of the resulting texture, we apply manual retouching on the separated layers. We also crop the bottom of each layer such that it is flat to avoid unnecessary data.

Palette simplification and editing. We reduce the input colors to create the pixelation effect. We use the k-means clustering algorithm to extract dominant colours. We empirically found that eight dominant colours are usually sufficient to describe a landscape photograph and set this as default for k. The colors are then reduced by mapping them to the colors of the palette using the k-means labels. Apart from color simplification, palette extraction also opens up the possibility to change the output colors via GUI input. We show a recolored example in Figure 3.

Naïvely changing and reducing to the colour palette, however, tends to produce "salt and pepper" noise which degrades the quality of our outputs. We therefor apply a bilateral filter to reduce the noise without blurring the key features of our image. The images then get scaled down to further cluster pixels and enforce the pixelation effect. We decide to use a scaling factor of 3 for our experiments, meaning that every 3x3 pixel window will be merged. Note that

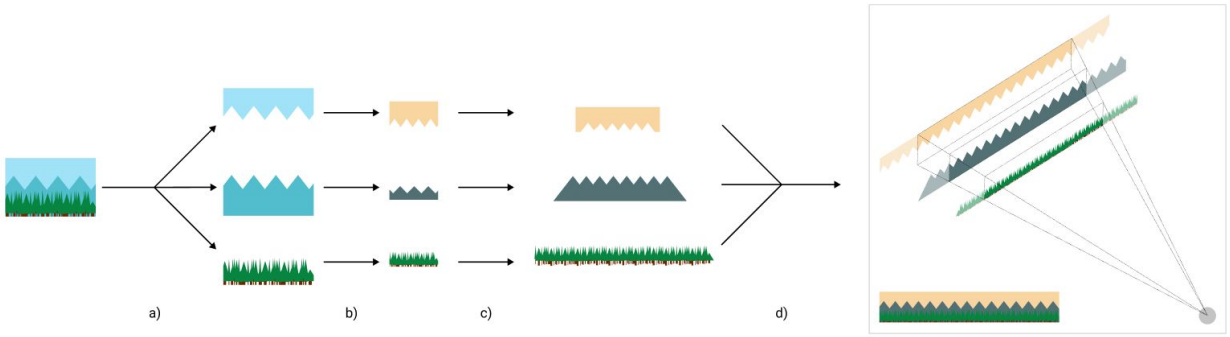

Figure 2: The main steps of our pipeline: a) Depth-based interactive segmentation. b) Down scaling and recolouring. c) Graphcut texture generation. d) Up scaling.

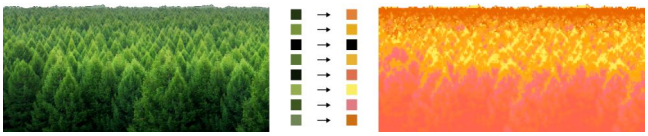

Figure 3: We show an example of our color reduction. The RGB input (left) is analyzed to retrieve the palette (middle). This can then be edited to produce the recolored result (right).

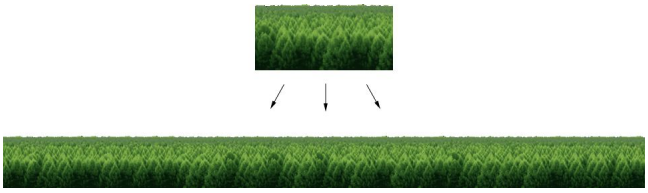

Figure 4: We show a texture generation example with the input segment (top) and the generated texture (bottom).

this parameter largely depends on the game design. We therefor encourage users to experiment different factors to suit their need.

Graphcut texture generation. We apply the graph cut based approach introduced by [Kwatra et al. 2003] to extend each layer to the required length. The input image is divided into patches which are then shuffled and recombined to produce an extended texture. Where patches overlap, the edges are adaptively changed in shape to create the most subtle edge transition possible. The visual quality can be influenced by adjusting the order and sizes of patches placed. Larger patches provide greater image coherency, while smaller patches produce a less repetitive result. We show a resulting texture in Figure 4.

Up scaling. The generated images are lastly up scaled to the target resolution which completes the pixelation process, and can then be imported into a game engine. We make use of the depth estimation again as a guide to position the layers in depth and control the parallax effect.

We show a final result in Figure 5. Our approach manages to create a convincing background. The mountain range and the trees are widely extended without visible seams or obvious repetitions.

Figure 5: We show a fully processed result with in-game geometry in the foreground.

The texture has an artistic, pixelated style without losing important features and thus keeps a realistic impression.

3 CONCLUSION

Our method successfully produces 2D side-scrolling game background from landscape photos. Our pipeline effectively lowers the entry barrier for game developers and changes "artist time" to computational runtime. Without prior artistic knowledge, users are able to produce believable backgrounds with relatively minor effort, making it a useful asset for independent game developers.

REFERENCES

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron F. Bobick. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM SIGGRAPH 2003 Papers* (2003).

S. Mahdi H. Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yagiz Aksoy. 2021. Boosting Monocular Depth Estimation Models to High-Resolution via Content-Adaptive Multi-Resolution Merging. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 9680–9689.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. 2020. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. (2020).

Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. 2021. Learning to Recover 3D Scene Shape from a Single Image. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 204–213.