Entertainment Computing (EC) Topics

Bachelor Informatik: Praktikum mit Bachelor-Arbeit, Softwarepraktikum

Master Medieninformatik: Praktikum 1 / 2, Masterarbeiten

Prof. Helmut Hlavacs

helmut.hlavacs@univie.ac.at size

http://entertain.univie.ac.at/~hlavacs/

Basis for many projects are the Vienna Vulkan Engine, the Adapted Vulkan Tutorial, Vienna Entity Component System and others. They can be found in

- https://github.com/hlavacs

The 2.0 version of the Vienna Vulkan Engine is currently being developed in the **nexgen** branch, but will be rolled out to the main branch before the semester start.

EC17 – Vulkan Imgui Interface Designer

Create an interface designer for Imgui that produces valid Vulkan code.

EC21 – Photorealistic Street Scene

Use the Vienna Vulkan Engine to create a photorealistic street scene.

EC22 – Graphical Vulkan Editor

Create a graphical editor for defining Vulkan based renderers. The output should be valid Vulkan code, e.g. for pipeline state objects, render passes, synchronization, threads, command buffers etc.

EC41 - Calibrate Colors in a 3D- Model

We scan an interior rooms using laser scanners and photos. The results are 3D-Models from large rooms. The colors depend on the current light situation, and might shift e.g. to blue.

The task is to use the available information (3D-model, color palette information, 360 degree photos) to undo this color shift, in order to get the original albedo (surface reflectance) for the colors.

EC46 – Denoising low-ray number path tracing pictures

Create a simple path tracer (offline). Implement a parameter n that determines the number of rays that are sent out per pixel. This n then determines also the noisiness of the result. Use a Deep Learning approach to denoise the pictures.

EC47 – Vulkan Pathtracer

Use Vulkan to implement a real-time path tracer.

EC49 – Virtual Car for Commercial

The idea is to create a 360 degree video from some cool environment, e.g. a canyon, mountains, etc. Then render a photorealistic car into it. This could be used for production of car commercials, where the actual cars are added to the videos in post production.

Film a 360 Video With drone, render car over IT for commercial.

EC53 – Game AI Toolbox

Take over an existing toolbox for testing game AI algorithms. It is currently based on OGRE/Java Script.

EC54 – Automatic creation of Behavior Trees for NPCs

Take over existing approaches for automatic behavior tree creation using goal oriented programming and optimization (e.g. genetic algorithms). Extend this approach.

EC56 – Enhance Vienna Vulkan Engine

Implement Deferred Lighting, PBR, Light Volumes, lens flares, ImGUI, SSAO, MSAA, bone animation, post processing, …

EC60 – Really Large Terrain Creator

Create a terrain generator for really really large terrains, using the Vienna Vulkan Engine.

EC61 – Terrain Modeller

Create a terrain modelling tool that uses Surface nets to let designers create simple terrains with surface nets.

EC64 – Fire and Smoke by Deep Learning

Create fires, smoke or explosions procedurally using deep learning.

EC65 – Deep Learning Video Codecs

First idea: train a DL GAN to create image sequences (motion compensated image differences) taken from a video. Use the last N frames to create the next frame. So instead of having a video file, you would actually store the GAN and the first frame to kick off the video sequence, the GAN would then produce the whole video!

Second idea: use an auto encoder to compress the information of a sequence of video frames, then reproduce the frames.

EC66 – Deep Learning based Texture Generation

Make photos of surfaces of stone, wood, anything with surface geometry. Make a deep learning solution to extract normal maps, bump maps, albedo.

EC68 – Create simple virtual robot actuator

E.g balancing something. Use small perturbations in friction, mass distribution to make the ML model robust. Use domain randomization (OpenAI grasping models)

EC74 – Game Engine Container

Design and implement a set of containers to be used in game engines. Containers should have either O(1) or O(logN) behavior.

EC76 – Animations in Vienna Vulkan Engine

Create joint/bone animations for VVE.

EC86 – Vienna Vulkan Render Lab

Design an abstraction library on top of the Vulkan Helper Layer to enable enable programmers to experiment with different Vulkan renderer strategies.

EC87 – Simple GUI for Vienna Vulkan Engine

Design a GUI for creating games with the VVE, similar to Unity, Unreal or Godot

EC88 – Simple Scene Editor outdoor

Use the Vienna Vulkan Engine to create an outdoor terrain editor based on marching cubes etc.

EC89 – Shader tool for engine

Use the Vienna Vulkan Engine to create a physically based shading tool.

EC91 – Vienna Game Metaprogramming Library

Create a C++20 metaprogramming library specifically for game engines.

EC92 – Performance of game engine containers.

Compare the performance of std::vector, slot maps, colonies with jump counted skiplists, and others.

EC93 – Dependency Graph for Games

Make a tool that lets you specify dependencies and resource usage (read/write) and that creates job schedules for the Vienna Game Job System.

EC94 – Algorithm to Code

Use a popular algorithm description form and create a way to create code directly from it.

EC95 – Work on the Vienna Physics Engine.

Implement more constraints, new models, create examples.

EC96 – Video Game Resolution Upscaling using Deep Learning

Use High Resolution game output, downscale it, then train a Deep Learning model to upscale the resolution back to its original resolution. Use it on different game content. Validate with validation set of games.

EC97 – Job System for GPUs

Extend the VGJS with a job system for GPUs using Vulkan Compute Shaders.

EC98 – Live Face Animation per DL

Use the same DL/autoencoder technique that is used to transfer faces from one actor to another to animate faces live.

EC99 – Game Engine Interface in VR

While in VR use the Vienna Vulkan Engine to create your scene. Use your hands to move stuff around, place it in the scene.

EC100 – Create a game with VVE and VECS

The Vienna Vulkan Engine (VVE) is a simple game engine. The Vienna Entity Component System (VECS) is a compile time ECS with high performance in mind. Create a game with them, e.g., by using OpenGL or Vulkan. Make use of VECS by including tens of thousands of objects.

https://github.com/hlavacs/ViennaEntityComponentSystem

EC101 – Copy realistic textures to segmented render output

Have a look at: https://isl-org.github.io/PhotorealismEnhancement/

Do something similar, try to get some results on it.

EC102 – Design and create a game that convinces vaccination sceptics to be vaccinated

Vaccination sceptics suffer from reactance, a condition that is created when they are pressured to do something and start resisting for fear of losing degrees of freedom.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4675534/

Design and create a game that uses established techniques to subvert this reactance and convince the players to be vaccinated: distancing, obfuscation, and mixing.

See https://ucloud.univie.ac.at/index.php/s/0yx0Mjn9e6itndl

EC103 – Assistive Technology for Mobility

Eine Flughafen-AI könnte blinde Menschen per Kamera entdecken und ihnen über eine App Anweisungen geben, wie sie zu bestimmten Plätzen kommen: Toiletten, Flugsteig, Restaurant. Uhrzeit ansagen und sagen wann es Zeit wird zum Flugsteig zu gehen.

EC104 – CPU based Rendering

Modern CPUs provide many cores, which can run threads in parallel. It stands to hypothesize that this might be already enough to 100% render a game just on this cores, without the GPU involved. Develop a software based renderer that uses the VGJS system to parallelize render tasks.

EC105 – Gambling Addiction Therapy

Develop a VR gambling machine simulation that enables people with gambling addiction to gamble in a safe space where they actually cannot lose money.

EC106 Self-cutting simulation

Develop a VR simulation for teenagers who suffer from anxieties and depression to such an extent that they start cutting themselves. Cutting should be done in VR to let of steam.

EC107 – Model NPCs with Machine Learning

Create an arena where NPCs fight against each other. Train an ML model (neural networks) to create efficient NPC behavior. Create a decision tree that explains this behavior as white box model.

EC108 – E-Sports Trainer

Create a tool that imports e-Sports log files that show team movement and tactics. The tool should 1) offer automated support explaining why a team lost or won, and 2) offer advice e.g. using simulation what the losing team should have done better in order to win a frag or the whole game.

EC109 – VECS Performance

The Vienna Entity Component System (VECS) is a compile time ECS with high performance in mind. Use the ECS benchmark to compare its performance to its competitors, especially entt

https://github.com/hlavacs/ViennaEntityComponentSystem

https://github.com/abeimler/ecs_benchmark

EC110 – Barrierefreie Uni Wien

Untersuchen sie möglichst viele Webseiten der Uni Wien auf Barrierefreiheit

EC111 – Lockless Data Structures

Work on lockless data structures in multithreaded games: queues, stacks, arrays, maps

EC112 – Detect Hands

Train a CNN to detect hands and hand poses.

EC113 – Skin Doctor App

Create an application to help skin doctors to create skin maps of patients.

EC114 – GANs of Stuff for Videos

Create GANs to create stuff, like cars, people, houses, trees, … Use them to compress videos with these objects.

EC115 – Deep Learning for Face Expression Identification

In VR the visor covers the upper part of the face. If the user is represented by an avatar that mimics the users facial expression, a camera cannot detect the upper part of the face. Train a NN that takes in the mouth/lower face parts and identifies the expression of the upper face part. Use videos of talking faces.

EC116 – Digital Human Pipeline

Human digitization pipeline. Use the latest iPhone to create a 3D model of human beings.

EC117 – Soccer Player Pose Estimation

Estimate the poses of soccer players in live TV.

EC118 – Detect threatening poses on Video surveillance

Assume you have video surveillance footage, detect poses that indicate threats or attacks.

EC119 – Identify Influence Factors for Object Detection

When training a CNN or using YOLO for finding objects (e.g. a hand) you have to apply many variations to the images to generalize the concept hand, like random backgrounds, different shadows and lights, blocking etc. The question arises how big the influence of each variation type on the detection rate is? Use a hand model in Unity, create variations, train a CNN, then detect hands. Try many combinations of variations and evaluate the detection rate. Infer the influence of each variation.

EC120 – DAG for a Job System

In a video game, systems need access to resources. Reads can be done in parallel, but writes must be exclusive. Design a scheduler that accepts functions + a list of accesses to resources. The scheduler then creates schedules (Directed Acyclic Graph, DAG) where the functions can work in parallel without interfering with each other.

EC121 – Outpatient Scrap Book App

In order to long term motivate patients to communicate with their clinicians, we envision an app that can be customized by patients like a personal scrap book. The app offers empty pages that can be filled with Infos, medical drug info, yoga, positive psychology, personal music, drawings, quizzes, boxing or other physical motions to alleviate anxiety and anger, fotos and videos, etc. The app is not only a source or information and patient communication but also documents the patient journey.

EC122 – Stage Simulator

Modern stages for live music must be able to display numerous effects, like video wall, light, smoke, fire, confetti, etc. Create a simulator for stage effects using a game engine.

EC123 – Soft Body Simulation

Create a simple soft body simulation for the Vienna Vulkan Engine. Soft bodies are 3D bodies that can be squeezed, like a stuffed pet animal made of clothes.

EC124 – Hair Simulation

Create a hair/fur simulation using the Vienna Vulkan Engine.

EC125 – Procedurally generated environments for First Responder Training

Procedurally generated Indoor Environments for Police and First responder VR Training. Indoors of houses, apartments, stairs, rooms, etc. Place the target/victim there to find (neutralize) it.

EC126 – Create the game: Herbert goes shopping

Characters: Herbert, typical western consumer, Rudi the shellfish

Players: single player (Herbert) or cooperative (Herbert, Rudi)

Story: Herbert has to satisfy his needs by shopping, Rudi has to eat stuff in sea, stuff may be healthy or poisoned.

Challenge and dramatic arc: go shopping, do not overspend budget, always new products from the industry.

Goals: satisfy needs by buying stuff for eating, drinking, entertainment, mobility, manage dopamine level. Rudi swims around and tries to eat healthy food, bust must stay alive. Food is marked as healthy or poisoned.

Resources: income of money, time, mobility, needs, dopamine level (normal, alert, critical)

Actions: Drive your super SUV, bike, or Toyota Camry to supermarket WALMORDOR, select products, balance price, dopamine boost, weight or products, importance of needs.

Rules: Cannot spend more money than we have. Cannot buy more than we can move in car or bike. The heavier the car the more poison goes into the sea. Dopamine level will make you dizzy and aggressive. Required dopamine level rises with more dopamine. Buy more unhealthy food in plastic, or entertainment products. The more you buy plastic the more you get products with plastic. The more products you buy the more you will need in the future.

Conflicts: The more plastic or oil consumed, the more the fish eat plastic and get poisoned, and dies.

Outcome: The game ends when all fish die or when all products are free of plastic.

EC127 – Habit Formation Toolbox

In this collaboration with the SFU you create a simple habit formation toolbox. Subjects carry out a simple task A (building, reading, collecting, achieving…) and randomly see a trigger, that lets them exhibit a pleasant surprise by doing a simple quick task B. This way we want to establish a pattern that leads to habits, and subjects would be trained to form them. In the first half of the experiment we would establish the habit and ask subjects if they are actually inclined to continue doing them.

In the second half we try out various inhibitors that try to interrupt the habit formation.

EC128 – Musik-Übungs-App

In dieser Kooperation mit der Musik Universität Wien erzeugen Sie eine App, die den Studierenden der Musik-Uni hilft, ihr Instrument zu lernen. Dazu gibt es schon ein Konzept, das sie in Kooperation mit Studierenden der Musik-Uni umsetzen.

In this cooperation with the Music-University Vienna you create an app that helps students to learn their instrument. There is a ready to be used concept for such an app, which you implement in a cooperation with music students.

EC129 – Vienna Physics Engine in SYCL

Port the Vienna Physics Engine to SYLC. The goal is that (parts) of it should run on the GPU.

https://github.com/hlavacs/ViennaPhysicsEngine

EC130 – Vienna Physics Engine in Vulkan Compute Shader

Port the Vienna Physics Engine to use Vulkan Compute Shaders

https://github.com/hlavacs/ViennaPhysicsEngine

EC131 – Detect Body Size from Foto

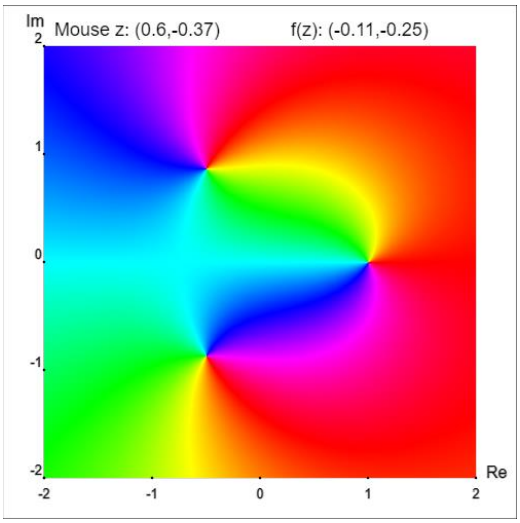
Use a foto to derive the size of human bodies. Also estimate their clothing sizes.

EC132 – Use NERFs and MERFs to render frames

NERFs or MERFs can be used to synthesize arbitrary camera angles from fotos taken from other positions. Use this technology to render frames in games and explore their potential for this task.

EC133 – Phase Portraits

Phase portraits (https://complex-analysis.com/content/domain_coloring.html) are colorful diagrams showing the phases of complex functions. Create a program that selects a complex function from a larger set of possible functions, and create its phase portrait in real time. Then select another function and use interpolation to move from the first function to the second. Keep on doing this to create a colorful, artistic visual experience based on mathematics.



EC134 – Trial Bike

Create a physics model for trial bikes. Create 2-3 simple levels for a game where you can drive trial bikes down a mountain ridge, or stairs, or make jumps.

EC135 – App for scanning license plates

Create an App that you can swing around, and it automatically finds cars, and reads the license plates from the cars. The result is a list of license plates that were read, together with the fotos where they were detected. You can also geo-tag the lace where the fotos were taken.

EC136 – Car physics model

Implement a physics model for cars into the Vienna Physics Engine. Focus on tires, grip, suspension, torque, braking.

EC137 – Motorbike physics model

Implement a physics model for motor bikes into the Vienna Physics Engine. Focus on the interaction between tires, grip, curves, and driver weight.

EC138 – Water physics model

Implement a physics model for water surfaces into the Vienna Vulkan Engine. Focus on light refraction, surface ripples, waves and interaction with solid objects, spray, Laplace equation, Navier-Stokes.

https://www.youtube.com/watch?v=kGEqaX4Y4bQ

EC139 – Ragdoll physics model

Implement a physics model for rag dolls into the Vienna Physics Engine. Ragdolls should be animated, or just fly on the floor. Animations should interact with the world, e.g. the floor, i.e. animations should be like motors, driving the joints.

EC140 – A Game Engine from Copilot

Can we use ChatGPT or Github Copilot to create a working game engine, just by typing commands into the prompt and creating source code? Try this to see how far we can go.

EC141 – Encoding Videos using Segmenting and GANs

Use Segment Anything to segment video frames, then train GANs to encode the segments efficiently.

EC142 – Wind for Soft Bodies

The Vienna Physics Engine contains soft bodies, create a scenario where simulated wind moves clothes.

EC143 – LLMs for NPCs

Explore the potential of LLCs like ChatGPT or Bard for steering conversations of NPCs in role play games.

EC144 – Minimal Vulkan Tutorial

Rewrite the Vulkan tutorial in such a way as to minimize the **number** of lines of code. Keep them under 500.

https://vulkan-tutorial.com/

https://github.com/Overv/VulkanTutorial

EC145 – Level Editor for the Vienna Vulkan Engine

Write a level editor for the Vienna Vulkan Engine.

EC146 – Finite Element Crash Model

Write a crash model for cars and other stuff for the Vienna Physics Engine. The crash model should simulate crashes, damages, things being deformed or destroyed. It should be based on Finite Elements.

EC147 – Building Destruction Model

For the Vienna Physics Engine, create a building destruction model. Buildings should be destructible, be destroyed by cannons, explosions, collapsing walls, etc. It should create dust, debris, wall parts, etc.

EC148 – Biped Walker Model

For the Vienna Physics Engine, create a model for a **bipedal walker**, i.e., a human walking on 2 legs. You can use the rigid body and constraint implementation in the engine. Use motors to act on the limbs. Start with a 2D version and some starting estimation for the values, then use continuous optimization to create ML models for each motor depending on the state of the walker. You can use ANNs, reinforcement learning or any other ML method. You can also make use of any python library.

EC149 – Annoying Slot Machines

Addicted gamblers playing on a slot machine often cannot stop until their money is gone. Slot machines could be mandated by law to implement annoying traits that ultimately keep gamblers away. A core feature of slot machines is the smooth progress of games – thus we envision a slot machine that stutters, meaning it shows increasing annoying time delays, depending on a specific player. Thus, if a player changes the machine, the time delays remain and get worse.

This is a cooperation with the SFU Vienna.

EC150 – Collision Detection for the Vienna Physics Engine

Implement curved objects (e.g., sphere, cylinder, …) for the Vienna Physics Engine. Also implement more collision detection algorithms, like GJK and Chung-Wang Separating-Vector.

EC151 – Planet simulation

For the Vienna Vulkan Engine, implement a planet scale simulation and rendering algorithm. This should heavily depend on LOD, and e.g. Perlin noise. You should see the planet from space, dice into its atmosphere, and land on the earth.

EC152 – Gaussian Splatting

Play around with https://github.com/graphdeco-inria/gaussian-splatting and create a couple of scenes. Mix in 3D content that is traditionally rendered. Render also onto a sphere to produce the contribution of the scene light and transfer the scene light onto the rendered 3D object. Effectively, you create a light probe.

EC153 – Full Jacobian Solver

Implement a Jacobian Solver for the Vienna Physics Engine.

EC154 – Featherstone

Implement Featherstone's Algorithm for the Vienna Physics Engine.

EC155 – Combine Python to the Vienna Physics Engine

Create a way to use the Vienna Physics Engine through Python.

EC156 – Move to Lockless Table

Reimplement the Vienna Physics Engine to be based on the Vienna Lockless Table.

EC157 – Renting Bias

Create a stealth serious game to uncover racist stereotypes for land lords not wanting to rent to people of color. Use the same approach as previous stealthy games like coming out or vaccine.

EC158 – Hiring Bias

Create a stealth serious game to uncover racist stereotypes for HR people not wanting to hire people from other ethnicities. Use the same approach as previous stealthy games like coming out or vaccine.

EC159 – Scripting an attack on the Battlefield

The conflict in Ukraine shows groups of soldiers attacking the front lines of their enemies. Create a battlefield generator where you can script such scenarios. In your scripting language you should be able to

- Create the battlefield
- Create defender lines, trenches
- Create attacking platoons, tanks, helicopters, planes, armed vehicles, Anti Tank Missiles, Anti Air missiles, artillery, drone surveillance and drone bobs, logistics
- Let attacks start, simulate the battles, determine wins and losses
- Simulate different weapons
- Show the progress in a rudimentary game world, does not have to be foto realistic

EC160 – A Cloud Gaming Engine

Create a cloud gaming engine, consisting of a server part that runs all the game logic, and a client part that downloads all assets, textures, and that receives rendering commands from the server.

EC161 – XR Battlecruiser Simulation

You are the captain of a battlecruiser in an epic space battle. You are hopelessly outnumbered by the enemy fleet and have to manage the resources of your cruiser. Resources include offensive (small fighters, canons, limited number of rockets, …) and defensive (deflector shields, repair crews and drones, force fields over holes, …), and energy. Player is wearing a Quest 3 headset with see through. In the room you mark areas with green screens, and the visor projects virtual screens on them. Like windows to space, tactical view, resource alerts, etc. In front of the player are desks with buttons and levers. Player can interact with the cruiser by pressing buttons and moving levers. These areas are shown in the visor via the cameras.

EC162 – VR Space flight

Player sits in a chair (with motors?), in VR he sits in a space fighter cockpit and can steer the fighter. The visor shows the space through a virtual window, and tactical information. Player can jump from system to system through portals. In each system the player can get a new quest, fight, mine resources, defend, make a fly by etc. Implement the concept and 1 or 2 quests.

Research question: Physical presence with or without vibrating chair or other cyber physical objects. Number of vibration motors, where they are attached.

EC163 – Image color filter for color blind viewers

Color blind viewers can often not distinguish between e.g. red and green. Create an image filter based on Vulkan Compute Shaders that takes in a rendered image and outputs an altered image where these colors that cannot be distinguished have been changed into one that can be better identified. At its core, it works like an Enchroma glass.

EC164 – Learn to give negative feedback

People with severe disabilities often struggle with giving negative feedback to the persons caring for them, in fear of being less liked. Create and evaluate app/web based technology that teaches these people to give negative feedback without anxieties. This is a cooperation with https://www.independo.app/

EC165 – Learn the concept of time

People with severe disabilities often have no concept of time, be it clock time or time duration. Create an app /web based technology that helps them to understand the concept of time, or at least to use it whenever they plan ahead for an upcoming day, or some activity later on this day. This is a cooperation with https://www.independo.app/

EC166 – Legolike Game for 3D training

Make a game like Lego. The purpose is to train understanding and mentally modelling 3D structures.

EC167 – Soccer/Football Trainer

Create a simulator that can take as an input a dynamic scene from a soccer/football game, and use Monte Carlo Tree Search and simulation to propose game play actions that lead to a goal or prevent a goal.

EC168 – Save Messenger based on Email

One important security issue for email is the possibility of communicating with anyone on the planet, and not distinguishing between ingroups and outgroups. Also the email address can easily be spoofed. Create a simple email client that makes sure that you only communicate within groups / your company / your employer, and which effectively blocks outside messages (irrespective of email sender address), and which prevents you from sending emails to any outsider. This effectively results in something like Slack, but on email basis. Also enable guarded curated communication with the rest of the world. You might enforce this with encryption, certificates, etc. on the client side, or guarded areas through the email server.

EC169 – Stiff Soft Bodies

For modelling feet, implement stiff soft bodies, which can deform a little bit, but may exert force like a rigid body.

EC170 – SpaceX like Rocket Landing

Create a simulation in a game engine (Godot) that lets you land a rocket in orbit safely on the earth's surface, just like SpaceX. Use a physics engine like the Vienna Physics Engine, create a model of the rocket and vectored thrust, fuel volume and mass, atmospheric drag, mass and inertia, force, torques, etc. First create a solution in 2D, then go on to 3D. Use a Godot planet creator to create random planets to visually simulate a planet, atmosphere, and surface.

EC171 – Realistic Limb Skin and Muscle Model

Body parts are usually modelled with rigging and bone animation. However, skin is usually not folded realistically, and muscles deform when moved. Create an arm/leg model using bones, muscles and a skin model that realistically folds skin and stretches muscles when the limb joints are bent. You can use the Vienna Physics Engine's clothing model to model the skin. (see e.g. https://kangaroo-builder.com/)

EC172 – Talk to the Lamp

Create a VR scene where you are a car driver in the city, and drive through the streets. Occasionally at crossings there will be dangerous situations: a child walks at the zebra crossing, you turn right and a cyclist comes along, people reading their phone cross without looking. Create intelligent traffic lights that project bright information onto the street to warn cars, cyclists and pedestrians of imminent danger. Show that this can improve safety.

EC173 – Neural Rendering

Create a diffusion model for rendering video game frames. It takes N previous frames + a currently created frame and outputs a full rendered frame. The input frame should be created using ray tracing in real time with a much smaller resolution. For training, also a full resolution version should be created. The rendered output in the game could be even just material and lighting data without color rendering. For training, the fully rendered high resolution output is necessary though.

EC174 – Remove light and shadow from 3D models

3D models that have been created from fotos or LIDAR scans will often show specular light reflections or shadows on their texture. Extend an existing thesis using new methods like diffusion models or transformers, or using light probes when taking the fotos.

EC175 – Extend the Vienna Game AI Library

The VIGAI library implements a number of algorithms. Extend the library with e.g. reinforcement learning, deep learning, MCTS, GOAP, Rule based systems, Blackboard Architectures, CNNs, etc.

EC176 – SPIR-V to Slang

Create an ML model that can translate SPIR-V to Slang / HLSL.

EC177 – Game Streaming with Vulkan Video

Implement Game Streaming with Vulkan Video into the new Vienna Vulkan Engine 2.0. Use Quantization maps to adapt the bitrate.

EC178 – Neural Texture Compression

Create a lightweight version of Nvidia Neural Texture Compression. It should be lightweight enough to allow decompression directly in the shader. Cut the texture into smaller tiles and create a model for each tile independently. This should result in smaller models and faster evaluation for each tile.

EC179 – Standard Vulkan Application

Vulkan suffers from too much freedom. Create a standard Vulkan renderer path that results in components that can be enabled or disabled, or even automatically created by LLMs.

EC180 – Upscaling in VVE 2.0

Create something similar to DLSS upscaling for the Vienna Vulkan Engine 2.0.

EC181 – Denoising in VVE 2.0

Create something similar to DLSS denoising for the Vienna Vulkan Engine 2.0.

EC182 – Deferred Rendering in VVE 2.0

Create a deferred renderer for the Vienna Vulkan Engine 2.0.

EC183 – Ray Tracing Renderer in VVE 2.0

Create a ray tracing renderer for the Vienna Vulkan Engine 2.0.

EC184 – Post Processing in VVE 2.0

Create a post processing path and several examples of post processing for the Vienna Vulkan Engine 2.0.

EC185 – Spec 1.1 - 1.4 for VVE 2.0

Implement major features for Spec 1.1 - 1.4 for the Vienna Vulkan Engine 2.0 helper layer.

EC186 – Streaming in VVE 2.0

Spec. 1.4 allows streaming of resources parallel to rendering. Implement this feature in the Vienna Vulkan Engine 2.0

EC187 – CPP for VVE 2.0

Port the VVE 2.0 helper layer to C++ using the HPP version.

EC188 – Console and GUI for VECS

The Vienna Entity Component System is available in v1.0. Create a console allowing for complex queries and a GUI showing which entities are currently in it for each running application. VECS instances can attach to the GUI using any IPC mechanism or simply using TCP sockets on local host.

EC189 – VVE 2.0 to Android

Being VVE 2.0 to Android.

EC190 – Render Graphs for VVE 2.0

Being render graphs to VVE 2.0, similar to those used in Blender.

EC191 – Improve the Adapted Vulkan Tutorial

The Adapted Vulkan Tutorial uses Spec 1.0 and C only. Add more code to show how it works with Spec 1.1 – 1.4, plus HPP bindings.

EC192 – Bring Hydra to VVE 2.0

The hydra scene manager is available for academic purposes. Make a path for VVE 2.0 to use Hydra.

EC193 – Shader Creator for Slang for VVE 2.0

Create a Slang shader creator and compile tool for VVE 2.0. Include the Vulkan Graphics Pipeline library for this, allowing for JIT pipeline creation.

EC194 – Augment the VVE 2.0

Implement compute shaders, AO and MSAA to VVE 2.0.

EC195 – A remote rendering Client for Vulkan

Use libGPULayers to Catch vk commands and send them over Network to a renderer client. Use wrappers for memcpy to deal with mapped memory.

EC196 – Parallelize the Vienna Entity Component System VECS

The Vienna Entity Component System VECS is an ECS similar to Unity DOTS, storing its data in cache optimal archetypes. So far its programmed for sequential access only, but there is a simple concept for available for enabling parallel access. In this project you would implement this concept and make VECS parallelizable: https://github.com/hlavacs/ViennaEntityComponentSystem

EC197 – Implement a scripting Layer on top of the Vienna Vulkan Engine

Implement scripting ontop of VVE, like Python, TCL, …

EC198 – Rework an App about practicing Music

We have an app that schedules practice work for music students. Take it over and add new features. This is a collaboration with the University of Music Vienna.

EC199 – Implement Slang Reflection in Vienna Vulkan Engine

Slang offers a wider range of shader reflection mechanisms. Implement them into the Vienna Vulkan Engine.

EC200 – Implement Query Caching in the Vienna Entity Component System

The Vienna Entity Component System needs efficient queries, which can be sped up by using query caching. Caching is already implemented but it's not switched on and needs to be optimized and tested.

EC201 – Sweep and Prune

Implement sweep and prune broad phase algorithm. 1. Regularly choose the world axis with largest spread. 2. Use positions of all objects to find the axis with largest spread using PCA. Compare 1 and 2.

EC202 – Game Assets AI Generator

Integrate a downloadable LLM into the Vienna Vulkan Engine. The interface should be a text prompt. The AI should create models that are stored as glTF or PNG, WAV, etc. on disk, and that then are loaded into the engine and displayed. A solution could also provide a GUI with special tools for each asset type, or route prompts to different tools per type.

EC203 – Collision Detection for the Vienna Physics Engine on GPU

Implement a Vulkan compute shader that is able to quickly find possible collisions for the VPE.

EC204 – LUMEN like Global Illumination in VVE 2.0

Create a LUMEN like Global Illumination renderer for the Vienna Vulkan Engine 2.0.