Exploring the Depths of The Solar System

Petr Ershov
Technical University of Munich
Chair of Human-Centered Technologies for Learning
Munich, Germany
ershowpeter@gmail.com

Lukas Postulka
Technical University of Munich
Chair of Human-Centered Technologies for Learning
Munich, Germany
lukas.postulka@gmail.com

I. Introduction

The objective of our project is to create an educational virtual reality (VR) game that provides players with accurate and engaging insights into the planets Mars, Venus, and Mercury as only these planets have a hard surface the player can visit. The game is designed to offer a realistic representation of these planets, correcting common misconceptions and enhancing the player's understanding of planetary science without the need for traditional textbooks.

A key aspect of our design philosophy was to move away from fantasy-driven depictions and focus on factual accuracy. For instance, Mars is often depicted as a purely red planet, but in reality, it has a more complex color palette, ranging from yellowish-grey to sandy hues with a touch of red. By addressing such misconceptions, our game aims to provide players with a true-to-life experience of what these planets actually look like, fostering a deeper appreciation and understanding of our solar system. This approach ensures that players gain genuine knowledge about planetary characteristics and phenomena, making the learning process both immersive and educational.

II. Level Design

The level design for our VR game involved creating detailed and realistic environments for Mars, Venus, and Mercury. This process required careful selection and integration of assets, as well as construction of both surface and space levels to provide an authentic experience for the players.

A. Mars Level

For Mars, we utilized an HDR 360-degree real image captured by the Curiosity rover to create a highly realistic background. This image provided a genuine depiction of the Martian landscape, allowing players to feel as though they are truly on the surface of Mars. For the terrain itself, we used 3D models generated from 75 pictures taken by another Mars rover. These models enabled us to construct a lifelike Martian surface without the need for height maps.

To enhance the educational aspect, we placed typical Martian objects throughout the level. These included basalt rocks, which are commonly found on Mars, and models of Mars rovers sourced from the official NASA website. By incorporating these elements, we aimed to provide players with a comprehensive understanding of the Martian environment and its geological features.

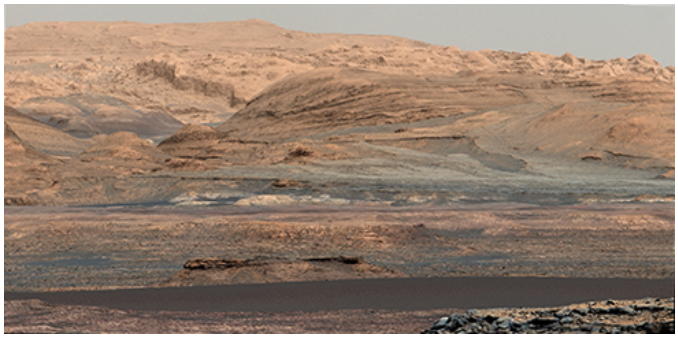

Fig. 1. Real images of Mars

Fig. 2. The Mars level in the game

B. Venus Level

The Venus level was based on real images taken by the Venera 13 probe.3 We strived to replicate the planet's thick, compressed atmosphere by incorporating dense clouds and fog into the environment. The surface of Venus, known for its previously active volcanic activity, was depicted using volcanic rocks scanned from Iceland - one of the most volcanic active places on earth. These stones were carefully placed to simulate the planet's volcanic terrain accurately.4

To further enhance the realism, we utilized textures and models that closely matched the features captured in the

Identify applicable funding agency here. If none, delete this.

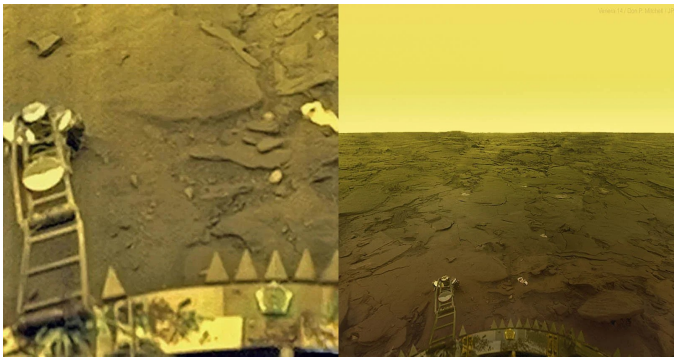

Fig. 3. Real images of Venus

```
void AAbstractItemActor::Interact_Implementation()
{
    if (InteractionSound)
    {
        ACPP_PlayerPawn* PlayerPawn = Cast<ACPP_PlayerPawn>(UGameplayStatics::GetPlayerPawn(this, 0));

        if (!PlayerPawn)
        {
            UE_LOG(LogTemp, Error, TEXT("Player character is not of type ACPP_PlayerPawn!"));
            return;
        }

        UAudioComponent* SFXAudioComponent = PlayerPawn->SFXAudioComponent;

        if (!SFXAudioComponent)
        {
            UE_LOG(LogTemp, Error, TEXT("SFXAudioComponent is null on the player character!"));
            return;
        }
        if (PlayerPawn->canListenToSounds)
        {
            SFXAudioComponent->SetSound(InteractionSound);
            SFXAudioComponent->Play();

            PlayerPawn->canListenToSounds = false;
        }
    }
```

Fig. 4. The Venus level in the game

Venera 13 images. This attention to detail ensured that players would experience an accurate representation of Venus's harsh and dynamic surface conditions.

C. Mercury Level

For Mercury, we drew inspiration from the Moon due to the similarities between their surfaces. However, we also emphasized Mercury's unique characteristics to provide a distinct and realistic depiction. It was important to create a surface that resembles the real planet. 5 and 6 Mercury lacks an atmosphere allowing the user to clearly see the stars and the stark, cratered landscape.

Typical objects such as ice, messenger parts and impact rocks were placed to highlight Mercury's geological features. These objects helped convey the planet's history of collisions and extreme temperature variations, contributing to a more immersive and educational experience.

D. Space Level

The space levels, where players choose which planet to explore, were designed with a focus on high-quality visuals and user experience. We utilized 8K textures for the planets, applied to spherical models, to ensure that they appeared detailed and lifelike from a distance. The spaceship itself

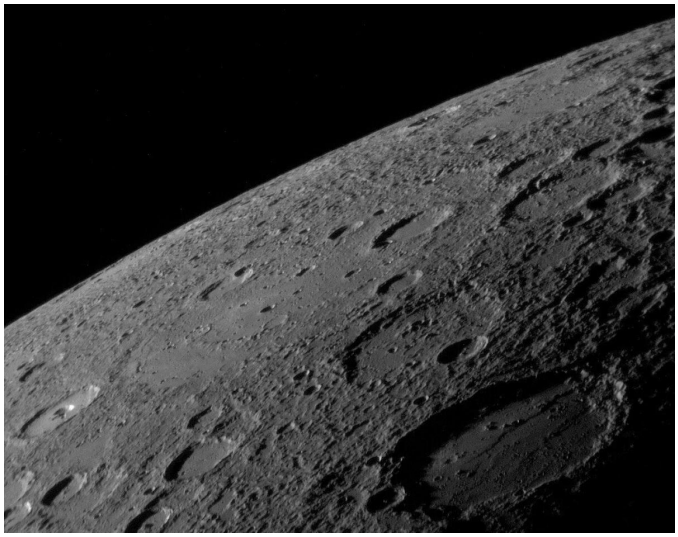

Fig. 5. Mercury real surface

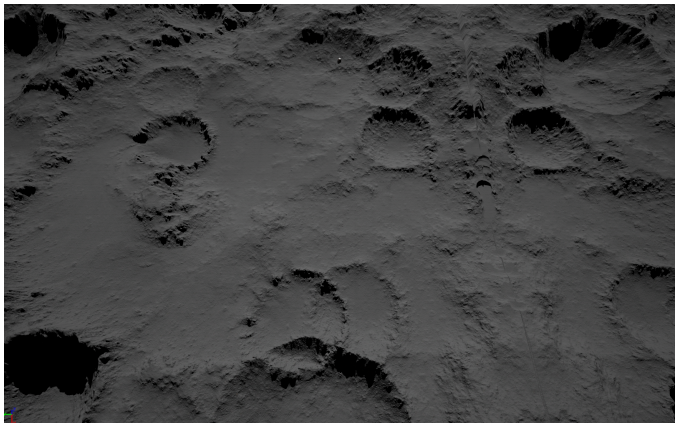

Fig. 6. Mercury game level

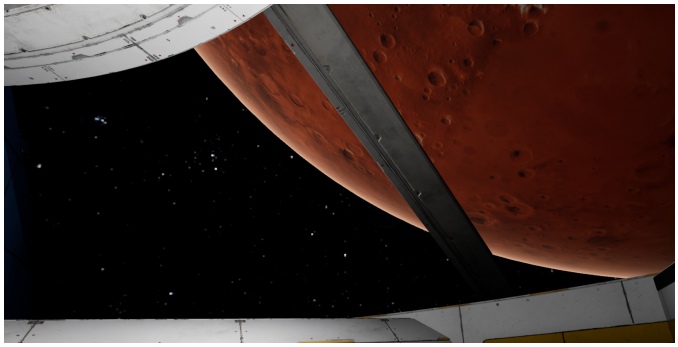

Fig. 7. Player approaching mars

was designed with panoramic windows, providing players with expansive views of the planets as they approach them.

By combining these design elements, we aimed to create an engaging and educational experience that accurately represents the unique characteristics of Mars, Venus, and Mercury. This approach not only enhances the player's immersion but also ensures that they gain a deeper understanding of these fascinating planets.

III. Design Aspects

A. Virtual Spaces Design

In designing the virtual spaces, we prioritized realism and immersion to create an authentic educational experience. Each planet's environment was meticulously crafted to reflect its unique characteristics, using high-resolution textures, real images, and scientifically accurate models. This approach ensures that players not only enjoy the visual aesthetics but also gain a deeper understanding of the planetary features and phenomena.

Again, the thick, oppressive atmosphere of Venus was simulated with dense clouds and fog, while the barren, cratered surface of Mercury was depicted with sharp, detailed textures. Mars, with its varied terrain and distinct color palette, was recreated using actual rover images and 3D models, ensuring a faithful representation of its surface.

B. Game Flow

The game flow was designed to be intuitive and engaging, guiding players seamlessly from one learning experience to another. Players start their journey in a spaceship, where they can view the earth from space. The player can choose now which planet he wants to explore by grabbing and holding a planet. (9) Once a planet is selected, the player approaches the planet (7) in the spaceship with panoramic view of the planet allowing the player to feel like it is in real life after that the user can choose to land on its surface and can walk around, interact with objects, and listen to informative narrations.

Throughout the entire game, a narrator accompanies the player, providing guidance and insights. The narrator is present when approaching a planet (7), during the landing sequence, and while interacting with objects on the planetary surface. This continuous narration ensures players remain engaged and informed, with the game flow supporting a natural progression of exploration and discovery. The transition between space and planetary surfaces is smooth, maintaining immersion and continuity, further enhancing the learning experience.

C. Learning Aspects

The educational goal of the game is to provide accurate, non-fictional information about Mars, Venus, and Mercury. By using real images and data, the game corrects common misconceptions and offers players a true-to-life understanding of these planets. Our aim is to replace outdated or incorrect knowledge with scientifically accurate information, presented in an engaging and accessible format.

One of the most prevalent misconceptions we addressed is the depiction of Mars. Often shown as an entirely red planet in popular media11, Mars in reality displays a more varied color palette.1 While it does have reddish hues due to iron oxide on its surface, it also features yellowish-grey and sandy areas. Our game captures this complexity, providing players with a more accurate visual representation of the Martian landscape. This correction helps dispel the oversimplified image of Mars and fosters a deeper appreciation for its true geological diversity.

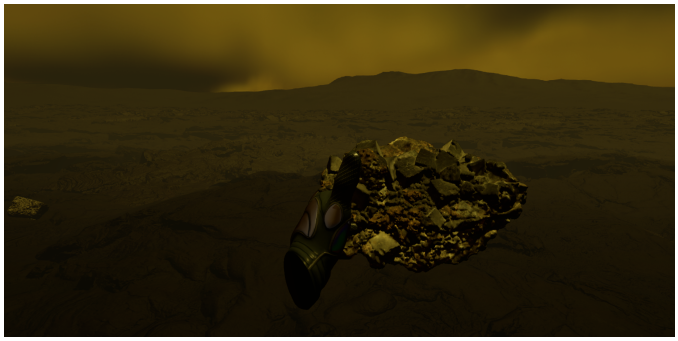

Fig. 8. User holding and listening to information about this object

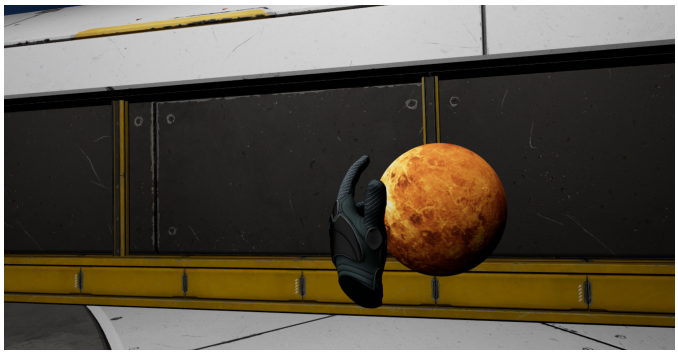

Fig. 9. Player choosing to go to Venus by grabbing the planet

Similarly, Venus is frequently portrayed with dramatic lava rivers and constantly erupting volcanoes.10 While there is evidence of relatively recent volcanic activity on Venus, it is not as widespread or visually dramatic as often depicted.3 Our game presents Venus with its thick, oppressive atmosphere and volcanic rocks, accurately reflecting the planet's true state. By correcting these exaggerated portrayals, we provide a more truthful understanding of Venus's geological history and its current environmental conditions. Interactive elements, such as picking up rocks or examining rover models, are paired with narrated explanations to enhance learning. This approach ensures that players are not only entertained but also gain substantial knowledge about planetary science.

IV. Technical Structure

A. Player Pawn

The VRPawn class is used to get hold of players hands and allow grabable logic. The grabbing itself is done via collision of hands mesh and one of specific meshes. It also contains fading logic and music components that are assigned a playing sound. More about those topics in the following sections.

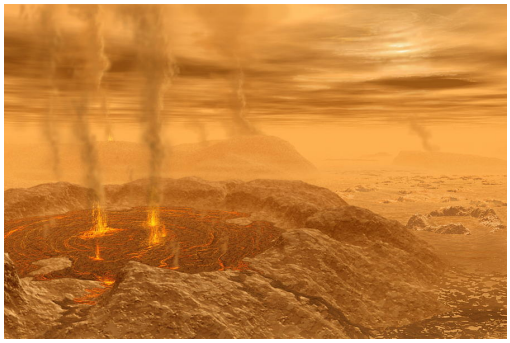

Fig. 10. Google search result for ”Venus Surface”

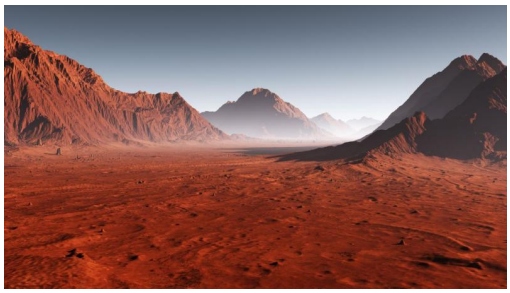

Fig. 11. Google search result for ”Mars Surface”

B. Interaction Design

On Figure 12 you can find the corresponding UML diagram of Interactable actor implementation. (Please not that many technical aspects of Unreal Engine and Blueprints like Events and casts are left out here for ease of reading). Whenever on of player's hand is colliding with objects that could be casted to BPGrabbable or it's subclasses, event is being triggered on the object. The BPGrabbable logic only assigns it's sound

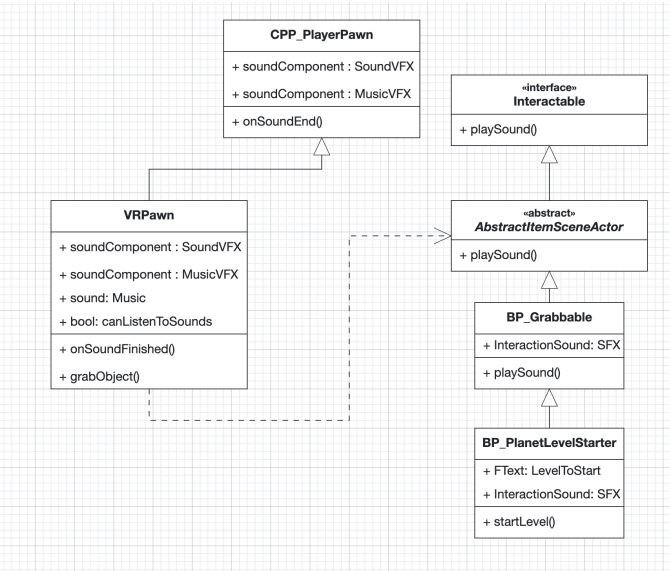

Fig. 12. C++ UML For Interactables

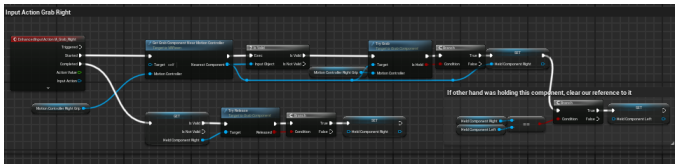

Fig. 13. Grab Logic

```
void AAbstractItemActor::Interact_Implementation()
{
    if (InteractionSound)
    {
        ACPP_PlayerPawn* PlayerPawn = Cast<ACPP_PlayerPawn>(UGameplayStatics::GetPlayerPawn(this, 0));

        if (!PlayerPawn)
        {
            UE_LOG(LogTemp, Error, TEXT("Player character is not of type ACPP_PlayerPawn!"));
            return;
        }

        UAudioComponent* SFXAudioComponent = PlayerPawn->SFXAudioComponent;

        if (!SFXAudioComponent)
        {
            UE_LOG(LogTemp, Error, TEXT("SFXAudioComponent is null on the player character!"));
            return;
        }
        if (PlayerPawn->canListenToSounds)
        {
            SFXAudioComponent->SetSound(InteractionSound);
            SFXAudioComponent->Play();

            PlayerPawn->canListenToSounds = false;
        }
    }
```

Fig. 14. Sound Setting

property to Pawn's MusicSFX Component which if possible plays the sound. Boolean property allows us to hear only one sound at once(Queuing was also considered but it makes it uncomfortable as some sounds take much time to play). Another inheretence of abstract item actor is a planet level starter, these are the planets that can be seen on the diagram. These also have a logic of starting levels whenever picked and hold for (SoundDuration - 3 seconds) time for user to hear the whole intro and start into the level.

C. Controls and UI

Basic movement and rotation are done via teleportation logic in VRPawn and allow user to teleport via right hand stick and rotate vie left hand stick. For an in-game explanation two user widgets are created in the menu and on the first solar level(Earth). The widgets also include videos that are created via an interesting play around with materials. Starting menu level gets hold of players point finger direction, linearly interpolates a collision line in forward direction and changes the input class to another one, allowing us to launch events via right hand trigger whenever this line collides with a widget button. These events either launch a solar earth level on the space ship or exit the game. Another interesting topic is Fading in and out of the levels(Figure 15). As we are in VR and no 2D screen is available to make it just black over time we need it to be a 3D object and be created around player's head. So it was decided for one two-sided sphere that appears in camera position over a timeline of 72 frames(around 1 second) and changes it's transparency each frame.

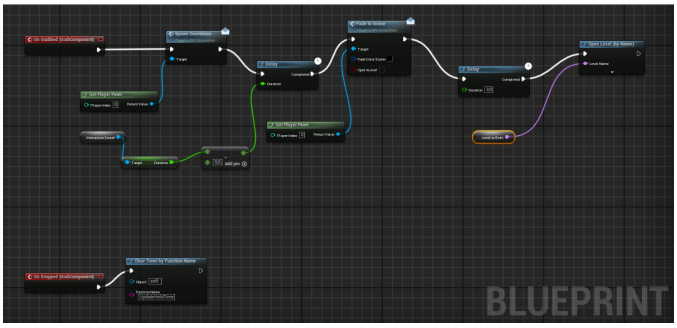

Fig. 15. Fade In

```cpp
// Called when the game starts or when spawned
void ACPP_SpaceShip::BeginPlay()
{
    Super::BeginPlay();
    // Find the first TargetPoint in the level

    TArray<AActor*> FoundActors;
    UGameplayStatics::GetAllActorsOfClass(GetWorld(), ATargetPoint::StaticClass(), FoundActors);
    if (FoundActors.Num() > 0)
    {
        TargetLocation = FoundActors[0];
    }
}

// Called every frame
void ACPP_SpaceShip::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

    // Move towards the target location
    FVector CurrentLocation = GetActorLocation();
    FVector Direction = (TargetLocation->GetActorLocation() - CurrentLocation).GetSafeNormal();
    FVector NewLocation = CurrentLocation + (Direction * MovementSpeed * DeltaTime);
    SetActorLocation(NewLocation);

    // Check if the planet has reached the target
    CheckIfReachedTarget();

}

void ACPP_SpaceShip::CheckIfReachedTarget()
{
    FVector CurrentLocation = GetActorLocation();
    float DistanceToTarget = FVector::Dist(CurrentLocation, TargetLocation->GetActorLocation());

    if (DistanceToTarget <= 10.0f) // Adjust tolerance as needed
        {
        // End the level
        UGameplayStatics::OpenLevel(this, FName(*GetWorld()->GetName()), false);
        }
}
```

Fig. 16. Space Ship Movement

D. Space Ship Movement

Allowing a player to fly around the solar system was a tricky task as planets and the whole solar system are done in their almost real sizes.(Figure 16). Whenever a static mesh moves at such speed it affects objects that are simulating physics and also would require the player pawn to be part of a space ship component which does not comply with many unreal engine laws. So the decision was done to make everything except the space ship to move. The planet and stars try to move it's center to the specific point on the scene. This logic is implemented in CPPSpaceShip which parents the BPPlanet class for the planets and sphere objects within the game.

E. Gravity

The Surface levels and Solar levels utilise Unreal Engine's physics gravity components to set it, e.g. Gravity on Mars

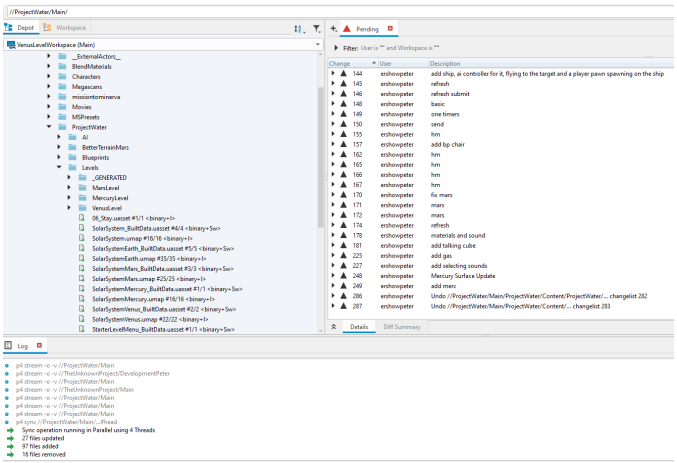

Fig. 17. CI in Perforce

is only about 38 percent of Earth gravity. Space levels are configured for zero gravity and objects behave realistically but for performance e.g. wind and obstacles are left out. You can always try to play with James Webb Space Telescope model on the space ship and see how it behaves.

F. CI

For continuous work on the project we used the Perforce Helix Core Server which runs on 2 GB Memory / 50 GB Disk / FRA1 - Rocky Linux Perforce Helix Core container(Figure 17)**??** as a Digital Ocean instance. Though it has quite a low memory, it allows to use an Unreal Engine with perforce source control, blocking the components(e.g. levels within the game) if those are being edited by another user. It helped us to avoid any single merge conflict and reverting of files. By expanding a memory to 16 GB Memory/ 250 GB Disk the server will allow Continuous Deployment by using scripts to e.g. cook and package the game for specific configuration. Currently the game is cooked and packed on USB storage and weights approximately 2.3 GB. Compatibility is confirmed for all Meta(Oculus) Quest devices and Varjo XR, but as it's done with OpenXR library and with Enhanced Inputs configured for devices supporting the library, it should be playable on all those devices.

REFERENCES

[1] https://science.nasa.gov/mission/messenger/
[2] https://science.nasa.gov/mission/mars-2020-perseverance/
[3] https://science.nasa.gov/mission/msl-curiosity/
[4] https://science.nasa.gov/mars
[5] https://science.nasa.gov/venus
[6] https://science.nasa.gov/mercury
[7] https://sputnikglobe.com/20171009/mars-mystery-origin-of-life-on-earth-1058073063.html
[8] https://arstechnica.com/science/2017/02/venus-computer-chip/
[9] https://www.esa.int/esatv/Videos/2018/10/Planet_Mercury
[10] https://sciencephotogallery.com/featured/surface-of-venus-chris-butler.html/
[11] https://stock.adobe.com/de/search?k=mars+surface/