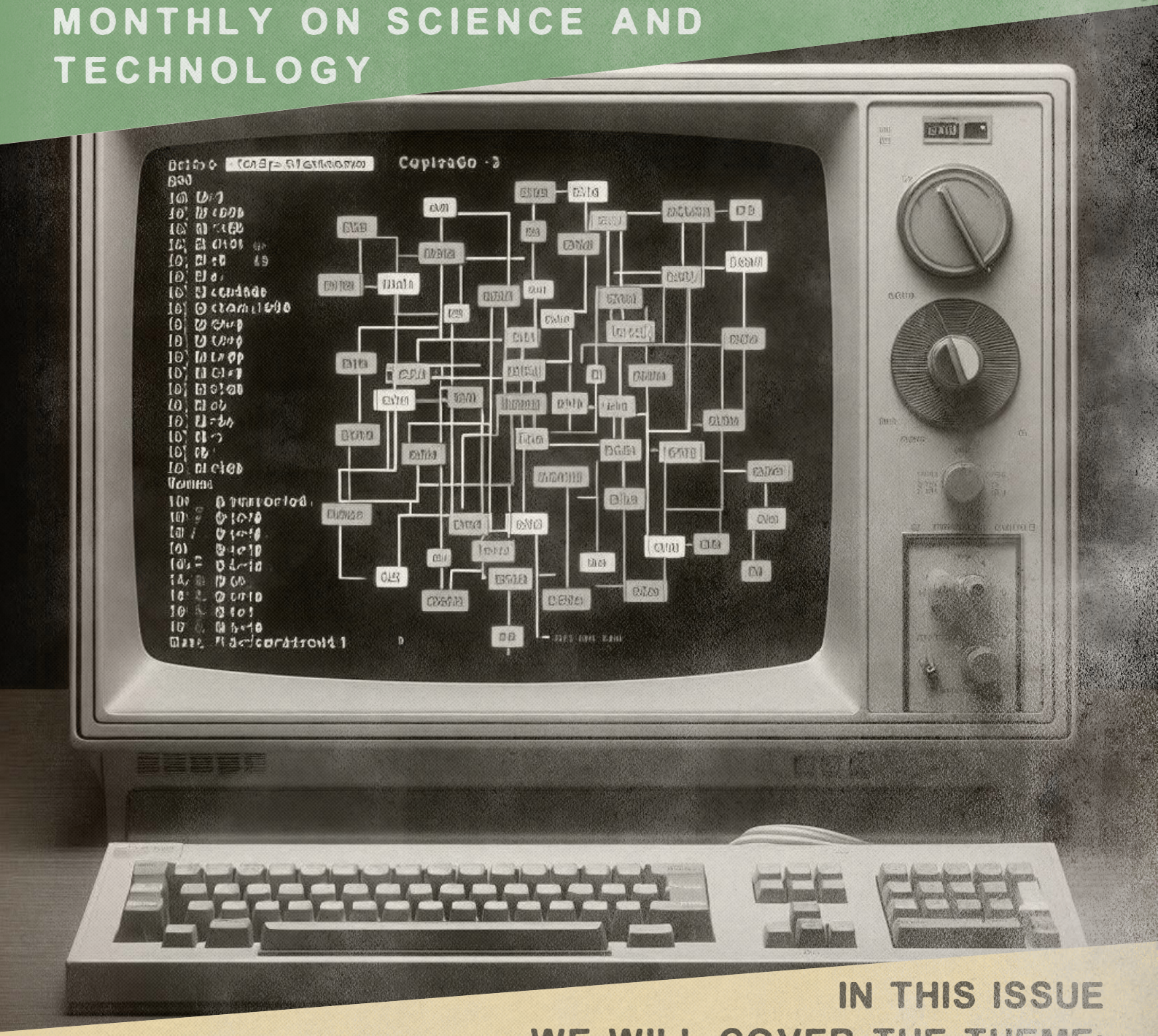
YOUNG RADIO AMATEUR

NUMBER 27

MONTHLY ON SCIENCE AND TECHNOLOGY



IN THIS ISSUE WE WILL COVER THE THEME:

HOW TO CONSTRUCT A COMPUTER GAME?

Engineers from the creative group **INTROVERT LEGION** will reveal what tools and instruments are suitable for successfully constructing a computer game. As an example, we will look at the computer game **ROCKTOPIA: INVADERS FROM THE PAST FUTURE**, which the authorial collective is diligently working on.

VOLUME 3, JANUARY 1973, PRICE 4 KČS

HOW TO CONSTRUCT A COMPUTER GAME?

Prepared by: Dipl. Ing. Judáš Novák, CSc.

I met with engineers from the creative group INTROVERT LEGION at the Milky Bar Star. We discussed how the computer game ROCKTOPIA: INVADERS FROM THE PAST FUTURE is created, on which the entire collective is intensively working. We were mainly interested in the technological aspect of the entire process - what new, modern tools they use, how they approach optimization, what technological challenges they encounter, and how it is with translation into foreign languages. You will learn everything in the interview. I wish you pleasant reading.

Article Author Meeting Engineers from INTROVERT LEGION at Milky Bar Star

Good day to you. Let's get right to it - as they say - from Adam. On what computers will it be possible to play your game?

Good day. Not only on computers but also on other devices. We call them platforms. We have them divided as follows:

Primary platform: PC; Steam and Epic Store
Secondary platform: Sony Playstation 5, Nintendo Switch 2 (unconfirmed), Steam Deck
Potential platforms: Xbox Series X

The game will support control via a game controller - gamepad. This will ensure the best gaming experience. There are many game controllers. Our game will support PC-compatible controllers for PS4, PS5, Xbox One, Xbox Series S/X, and Nintendo Pro Controller. If you want to play the game on a personal computer, it should meet the following requirements:

Operating System: Windows 10/11, 64-bit
Processor: Intel Core i7 - 7700, AMD Ryzen 5 1400
Graphics Card: NVIDIA GeForce GTX 1060, or AMD Radeon RX 580, or better. The graphics card memory should have a capacity of at least 6GB VRAM
Sound: Integrated sound card
Operating Memory: 8GB
Hard Disk: 20GB - this is an estimate

Tell us, what various programs and technologies do you use?

Engineers and artists from other production sections send their partial creations to a central development environment where everything comes together, and the game itself is created. This environment is the globally known program called **Unreal Engine 5** (currently in version 5.3.2.). We use this development environment in the form that it is available - without any custom modifications.

Why specifically Unreal Engine 5?

There are several significant factors that decided in favor of this development environment. They are:

- **Rendering Quality:** the development environment offers the best tools to achieve the quality of image, light, and shadows that we require.
- **Ease of Use:** more experienced engineers easily navigate this environment.
- **Visual Programming System:** the so-called Blueprint visual programming allows for easy and efficient creation of simple operations.
- **Previous Experience:** we do not hide the fact that the core of our art section has already used this environment in professional deployment in the creation of movie visual effects. They know how to work with cameras, with light, with coloring film, with creating complex film sequences.

For these reasons, the choice of Unreal Engine as the development environment was clear.

What software do your artistic workers use?

"The assembly" of the game takes place in the development environment of Unreal Engine 5. Here our artistic workers create unusual compositions from objects that were prepared in other programs (see below). They also set lighting and surface properties here. "They plant" grass, trees, shrubs, mushrooms. They place small stones, shape terrain. They create gorges, caves, mountains, meadows valleys. They control clouds, sun, fog. It is here that everything fits together, and our game world "comes to life." However, to have something to assemble, it is necessary to create objects. Currently, many objects can be purchased. We create all objects ourselves. We use the following tools:

Programs used by artistic workers:

Tasks:	Program:
Creation of standard three-dimensional objects (characters, stones, trees, shrubs, grass, etc.)	Blender
Creation of mountain ranges designed for the background	Gaea
Covering three-dimensional models with prepared surfaces and working with surface layers	Adobe Substance 3D Painter
Creating custom surfaces	Adobe Substance 3D Designer
Creating surfaces from photographs	Materialize
Creating components of the game's graphic user interface and production of some specific surfaces	Adobe Photoshop, Adobe Illustrator
Preparing materials for printing	Adobe InDesign (however, we also use ink pens, lettering templates, rulers, and French curves)
Creating custom fonts	Calligraphr, FontForge, Adobe Photoshop
Creating bones, attaching characters to skeletons, and animation	Blender

Supportive Plugins Used by Artistic Workers in Blender:

Tasks:	Supportive Plugin:
Batch Export of Models	Super Batch Export
Layered Animation	Animation Layers
Communication Between Control Skeleton and Shadow Skeleton, Animation Management	Frame Ranger
Basic Object Splitting	Noisy Cutter
Detailed Object Splitting	RBD Lab
Control and Adjustment of Surface Detail	Texel Density Checker
Heuristic Arrangement of Two-Dimensional Representation of Three-Dimensional Models for Optimal Use of Space Allocated for the Surface of a Given Object	UV packmaster
Custom Toolkit: Setting color management, working with vertex colors (randomization or based on UV), automated attachment of objects to the skeleton, automated attachment of the shadow skeleton to the control skeleton	Dino Tools

Software Used by Workers in the Game Design Section:

Tasks:	Program:
Creation of Game Environment Prototypes	Blender
Creation of Game User Interface Prototype	Adobe XD, Figma, Miro
Organizing Implementation of Game and Support Mechanics	Trello

Software Used by All Workers:

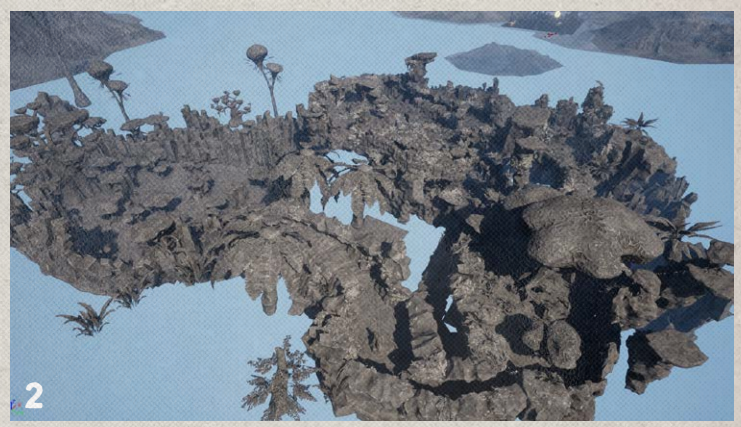
Tasks:	Program:
Data Storage and Synchronization Database	Synology Rackstation RS3614xs
Project Versioning	Apache Subversion, Tortoise SVN
Sharing Work Documents	Google Drive
Creating Work Documents	Google Docs
Communication Within the Creative Group	Skype

Supportive Plugins Within the Unreal Engine 5 Development Environment

Tasks:	Program:
Detailed Control of Sky Settings: Control of clouds, sun, day phase, intensity and temperature of lighting, as well as exposure. The system offers significant artistic control over the final output, providing a high degree of flexibility and many optimization options from a performance standpoint	Ultra Dynamic Sky
Tool for Creating Relatively Realistic Effects of Reactive Flora: Visual reaction of grass, shrubs, and trees upon physical contact with a character in the game. The potential of this system is not yet fully utilized. Full deployment will occur after the completion of work on the "vertical slice."	Prismatiscape
Proven Tool for Storing Data and Changes in Game Scenes: Used in games like Hydroneer, Forgive Me Father, or Night of the Dead - games very well-reviewed on the Steam platform	(EMS) Easy Multi Save

Well, that was certainly an exhaustive listing of the software equipment. Let's get back to your game - what is the brief process of creating what is called Level Design - environments in your game?

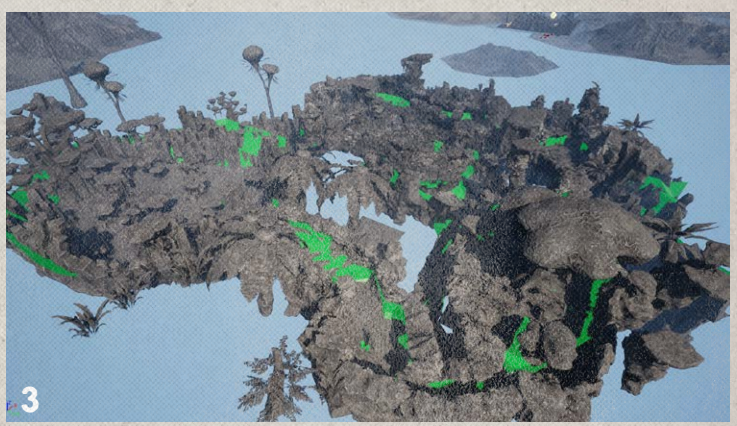
The process is systematic. Designs of spaces and environments gradually go through iterations from drawn concepts to a simplified three-dimensional model. This then moves into Unreal Engine, where it is used as a mockup for the production of game environments:



1 - Mockup - Simplified three-dimensional model

2 - Game environment created based on the mockup

3 - Mockup along with the game environment



Indeed, your environments seem quite intricate. Isn't that a big strain on computer performance?

Good observation. If we want the game to run smoothly even on less powerful computers, we must approach the creation of the game environment very sensitively and carefully. So, we are talking about optimization. In our game, we started optimizing right from the beginning of production. We knew that our environments would be full and detailed. We meticulously divide the environments into blocks, which we call "streaming volumes." This division ensures that those parts of the environment that the player does not see or cannot see are not rendered. Thus, they do not unnecessarily consume CPU performance or occupy space in computer memory. The functionality of these blocks is fully automatic - meaning that the game itself loads the data it needs at any given moment, while the player notices nothing - the gaming experience is thus smooth and uninterrupted.

However, this is not the only optimization tool. We also think of players who have really less powerful computers and would also like to play our game. Here we reach compromise solutions - by reducing the quality of certain parameters, we can achieve smoother gameplay. However, this is compensated by a degraded visual experience. These are the tools:

Flora Density: Reducing the amount of displayed grass, shrubs, mushrooms, and small stones
Shadow Calculation Accuracy: After optimization, the shadows that objects cast may be "jagged" or simplified in shape
Surface Calculation Accuracy: Reflections on surfaces may be simplified after optimization
Hiding Objects by Size and Distance: If small objects are in the frame at a distance from the player beyond a certain limit, they can gradually stop being rendered (the object gradually becomes transparent)
Reducing Surface Size: Smaller surfaces occupy less space in the computer's memory.

We know that the game is quite a busy affair. Constantly, various sub-programs and processes are running in the background. Does this also affect the overall performance?

Of course. As we mentioned earlier, the game environment is divided into blocks. Similarly, we have divided the launching of processes.
Local Processes: Processes that occur within active blocks (such as activating various objects)
Global Processes: Processes that occur continuously and are independent of which block is currently active (such as controlling lighting and atmosphere)

Interactivity - that's a new word in our parts. I've studied it - it's the system's ability to respond to user inputs and allow them to influence or control the course of events or content. What will interactivity look like in your implementation?

Nice definition. The trigger for interactivity in the game environment will mostly be our main character - the dinosaur. When our character approaches an object with which it can interact, the object begins to "communicate" with the character. Ways of such communication can be varied. Textual, light, sound, motion, or combinations thereof. In principle, however, they clearly indicate to the player what needs to be done - for example, which button on the controller to press. These are the basic types of interactions in our game:

Push: Moving objects to solve puzzles and facilitate movement through the game environment.
Grab-Carry: Carrying objects for a similar purpose, but there is the possibility to carry an object over obstacles.
Lever: Triggering and stopping mechanisms or changing two different states of mechanisms.
Button: Triggering and stopping mechanisms or changing two different states of mechanisms.
Pressure Plate: Triggering mechanisms by loading a pressure plate. The load can be the main character or some object.
Energy Emitter: Emits a beam in a certain direction. In the game, it is expected that the player will direct its path towards an "energy absorber" using an "energy ray bender."
Energy Ray Bender: See above, used to bend the energy ray.
Energy Absorber: The target into which the energy ray must be bent to trigger or stop some action.
Elemental Totem: At this totem, our character can exchange points collected during the game for some upgrades in three different categories.
Elemental Well: In this well, the transformation of our character to the next developmental stage takes place.

These interactions, more precisely, these interactive objects activate other objects that do not have their interaction logic. For example, doors - they can only be opened using a Button, Lever, Pressure Plate, or Energy Emitter. The player can thus interact not only with doors but also with various traps and mechanisms that spit lava or ice. Through interactive elements, the player can direct these objects or directly influence them so that they do not play against him but with him.

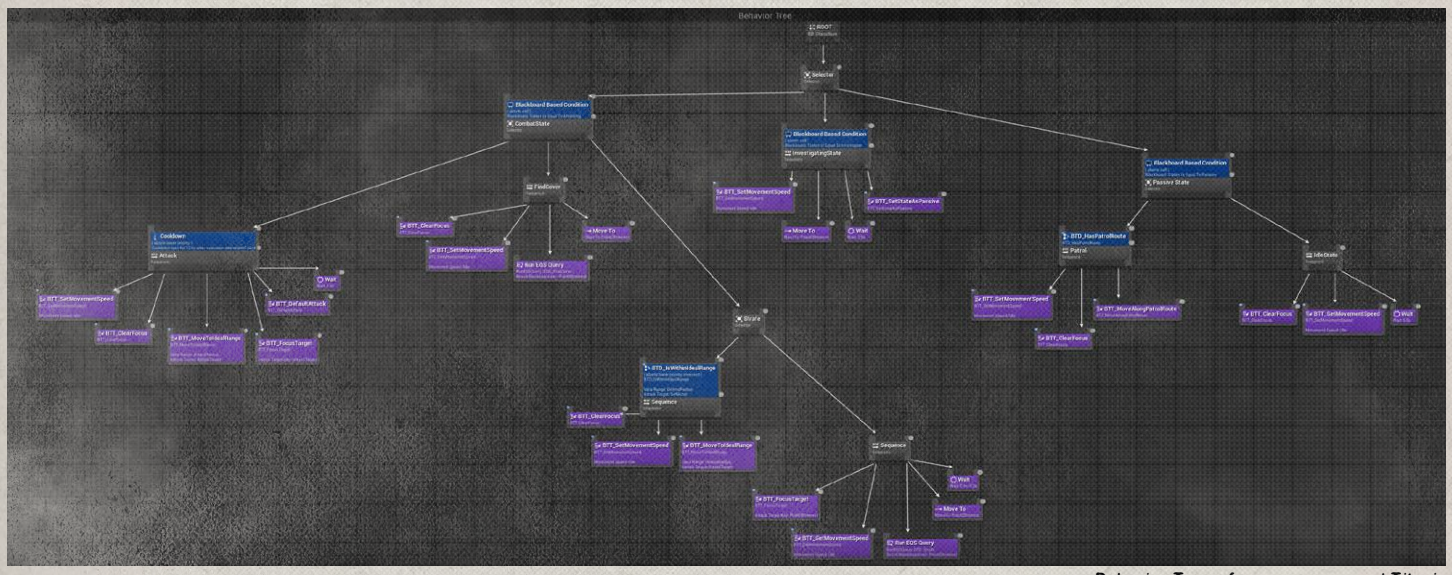
In one sentence, please - what are the main principles of the game?

The entire game loop revolves around finding food, and thus survival until the end, where the game sometimes helps the player achieve this goal, but other times it makes it more difficult.

About making the game challenging - I sense some overthought enemies here. Do these enemies have some sort of "intelligence"? Can they react to the main character in some way?

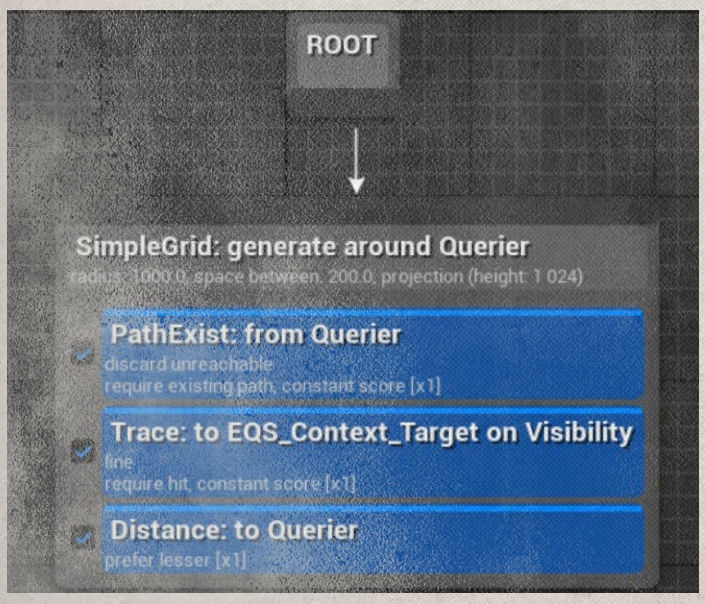
Certainly. We call it artificial intelligence. We use artificial intelligence modules that are integrated directly into the Unreal Engine development environment. For those more technically inclined, we could describe it as follows:

Artificial intelligence is based on various modules that work together. These include:
Behavior Tree: This module takes care of the logic of artificial intelligence.
Blackboard: This module carries variables that the Behavior Tree module exchanges with the surrounding world.
AI Controller: Besides basic detections, this module also manages the sending of data to the Behavior Tree and the Blackboard.



Behavior Tree of an enemy named Titanis

The Behavior Tree is a hierarchical structure of commands where each node signifies an action or decision. Instructions include commands like wait, attack, search, hide, and others. Within the Unreal Engine environment, we also utilize specifics of the game space for artificial intelligence needs. For example, the Environment Query System is used to find suitable hiding spots.



A layman's explanation of artificial intelligence could be given as follows:

The enemy sees you when you are close and within their field of view. The enemy can also hear you when you are further away and making noise. It is capable of pursuing you. You can hide from it. It will search for you for a while. If it doesn't find you, it will lose interest. However, if it detects you again—by sight or sound—it will start pursuing you again. If it is threatened, it can hide.

Do all the characters that the dinosaur in your game encounters try to attack it?

No, some serve as food. We have divided the characters into three categories:

Passive Side Characters: They do not attack, can be attacked but do not defend themselves. They serve as food.

Reactive Side Characters: They do not attack unless the player attacks first. They may or may not flee from the player.
Main Characters: Predators. They always attack as soon as they notice the player.

You mentioned that the enemy can hear the dinosaur. What kinds of sounds can it make?

Primarily, the sounds include footsteps, running, and jumping on various surfaces. The sound playback system monitors the type of surface the player's main character is moving on and decides which sound to play accordingly. In the Unreal Engine development environment, we can modulate aspects like volume based on the speed of movement and the distance from the player. We use footstep sounds for all larger characters. For smaller characters that may attack in groups, it would be disruptive to play all sounds for all animals simultaneously. In such cases, only the footstep sounds of characters closest to the player will be played. More distant characters will only produce secondary noises such as rustling grass, bushes, or very soft sounds of running on sand. Some characters will emit specific attack sounds,

like trumpeting or screaming.

An important part of the sound landscape includes ambient noises. These significantly contribute to the perception of the atmosphere. In the game, players might hear the wind in rock crevices, falling stones, geysers, or subthreshold sounds as a harbinger of an event. Sounds will be triggered by switches activated either by the main character or based on fulfilling conditions of a particular scenario. These switches are placed in the space during its construction.

A crucial component of sound is music. Only one musical theme will play at a time, dynamically transitioning to another based on the game situation, such as from a calm theme to a dynamic one.

Will it be possible to pause the state of the game and, for example, go to the Milky Bar Star and then continue playing the next day where we left off?

Yes. The Unreal Engine provides tools for saving the state of the game. However, we have chosen to use the "Easy Multi Save" plug-in, which enhances the game state saving options by adding additional custom parameters. The main advantages are:

- *Quick and efficient way to save and read complex game data.*
- *Practically unlimited number of save spots along with previews.*
- *Allows saving and loading states of various types of information (character components, the characters themselves, spatial blocks and subprograms, player status, or inventory).*
- *The module automatically decides which components to update or respawn.*
- *Multithreaded and deferred reading and saving of many actors.*
- *Clean file structure with support for file systems for PCs and console platforms.*
- *After updating the project, it keeps old saved files as relevant.*
- *Custom objects for saving settings or additional data.*

Telemetry - another foreign term and another definition: the collection of data for the purpose of monitoring, analyzing, and managing processes. Do you use telemetry data?

Telemetry is an important aspect in our field. It allows us to identify where players most frequently linger within the game environment, where they stop and do not proceed, or if they somehow exit the intended game area. This data can significantly help in identifying bugs and improving almost every aspect of the game, from playability through optimization to design. Currently, our game does not integrate any telemetry tools, but we plan to conduct several telemetry tests in the future using the product from Talos Interactive LLC, named Game Telemetry and Heatmap Recorder.

Last question. I assume that your game, ROCKTOPIA: INVADERS FROM THE PAST FUTURE, is not meant exclusively for our player, but that you intend to release it globally. This means the game will need to communicate with players in various languages...

The game will include several language versions. We plan to support: English, French, German, Spanish, and Portuguese. If we decide to support Asian languages, it will include: Japanese, Simplified Chinese, and Traditional Chinese.

Thank you for the inspiring interview and I wish you much success in your personal and professional future.

Thank you.