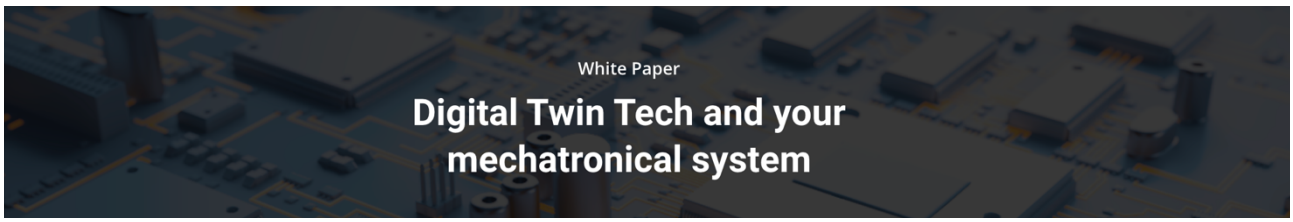


Introduction

In an ever-advancing world, digital technology allows you to realize a fully working digital version of your machine – a digital twin. Meet Prespective, the first digital twin platform, at the forefront of this disruptive development. Prespective enables you to develop and test your machine in the virtual world before the real machine exists. The platform is so interactive that it behaves exactly as the real machine would. Even the same control software runs on both digital twin and real machine. Unit040 is a company with years of experience and recognition in smart visualization of the high-tech industry. Come see the future in Prespective.

In this white paper, you will learn how to create such a digital carbon copy of your machine. As you might suspect, there is much more to it than we can cover in this text. It will take hands-on experience on how to perform all necessary actions. Learning by doing is the creed. You will notice with every new model that the steps will feel more natural and logical.
Unit040 organizes workshops where you can acquire the essential practical knowledge. Within four hours, participants will comprehend the basics of the workflow and get some tips and tricks. When you want more in-depth training, try the one or two day course and master all the ins and outs and do's and don'ts.
Next to these training options, Unit040 offers consultancy services to help you get started.

Unit040s mission and vision are that everyone should be able to profit from the benefits of digital twinning. Whether you are a rookie software developer or a seasoned mechanical engineer, you will most certainly find your own added value in a Digital Twin.
In that sense, you could compare Prespective to Photoshop. That image editing suite is cool for amateur photographers who want to improve their shots with some basic tools. But professionals appreciate all the options and features within Photoshop. You can go as deep as you like, both in Photoshop and Prespective.
At Unit040, openness is one of our core values and that's why we work to promote an inclusive and open framework within the Unity3D engine. Prespective is highly accessible. Even to the core, if necessary for super-advanced physics. On the other hand, you can just as easily take advantage of the embedded knowledge within Prespective and quickly build your Digital Twin based on predefined settings. Everything is possible; it is entirely up to you and your demands.

Chapter 1 – general overview

In the early days, system architecting was irrelevant. System development was merely a mechanical challenge, with camshafts dictating the motions in the machine. Then electronics took over, followed by software to boost intelligence even further. Suddenly, we could build smarter and more complex systems. On the downside, though, architects experienced a certain pillarization in their development teams: mechanical engineers vs. electrical specialists vs. software designers. Arguably, the biggest obstacle was that the latter of the three was always late to the party. They simply couldn't start writing code before the base of the machine had actually been built. Although a lot has improved in recent years, it's clear that there is still much to be gained.

The advanced machines engineers want to build nowadays, are getting ever more complex. So complex, in fact, that it is almost impossible to keep an overview. Too many intertwining details make them lose their grip on the requirements. To make matters worse, they have fewer resources at hand, both in terms of people and money, and they are pressed to raise the quality of the end product.

Another challenge is the wish for an agile design flow, with its short iteration cycles. Wouldn't it be great to watch the whole system improve every few weeks and show the progress to the customer? For that, you need virtual steel to build up your system, with virtual actuators and sensors, as well as a virtual platform to test the software. The industry has an ever-growing need for this, but the traditional tooling is lagging.

Evolution

The logical next step in system development – that also answers to the challenges mentioned above – is digital twinning. You have probably already heard the over-hyped stories, promising you the moon when you use the technology. But let us tone it down a little because digital twinning is no revolution; it's just an evolution from what used to be called virtual prototyping. For years companies have already been simulating their designs with FEM analysis or computational fluid dynamics, but always on the component level. This is very helpful, but the real challenge is in combining all components and simulating the whole machine. That means linking the tools from each development pillar, which can be a real Babylonian confusion of tongues.

A Digital Twin breaks down the pillars and unlocks the possibility of mixing various blood types. Prespective, our real-time simulation tool, has the ability to tap into all of the data models and sources that the different specialists are creating. It understands CAD models drawn in, for instance, Autodesk, Dassault or Siemens software. It accepts physical models made with tools such as Comsol, Matlab or Wolfram. The best part is that you can even run your embedded software on the Digital Twin created in Prespective, as it reacts precisely the same as the real system would.

With Prespective, designers with different backgrounds can create a common model so they can easily communicate with each other. Without the need for a complete restructuring of the development process, you will have a virtual product all your designers can work on in parallel, allowing them to create prototypes both fast and cheap. Just as easily, they can use it in their own department to virtually verify the assumptions they made in their model. And if they want, they can already start the basic iteration after the first design brainstorm.

Unity3D

Digital Twin tools come in two flavors. Major companies offer solutions, but these have the potential disadvantage of vendor lock-in. Although these brands claim to be open, their tools don't operate very well with third-party software. This is particularly impractical, as the different development pillars generally don't work with matching tools.

Next to the digital twin software from juggernauts, several speedboats are taking the industry by storm. Smaller contestants that are dedicated, adapt more quickly and are completely open. However, small certainly doesn't mean weak. When Unit040 started its Digital Twin journey about seven years ago, we faced the dilemma of developing an engine ourselves or using a strong existing base. It was a no-brainer that a dedicated development team of at least a hundred engineers was out

of our league. That's why we opted to build our Prespective platform on Unity3D, the biggest gaming engine in the world. Every day more than three thousand engineers are improving, updating and finetuning this engine.
Our digital twin platform is truly open. Not only does it accept input from a wide range of sources, but engineers also have access to the engine. By making modifications in Unity3D, designers can get a lot of help overcoming the hurdles they encounter when trying to solve very complex physical problems.

25 percent faster

Digital twinning is the modern way of prototyping. It's a new layer on top of your existing models, extending your possibilities. Estimated from real-life experiences, you can develop 25 percent faster with real-time simulation software like Prespective, all with less cost, fewer mistakes, smaller teams, fewer physical prototypes and fewer issues after commissioning. By providing insight into the requirements, you will drastically improve the communication with your (first tier) suppliers and, above all, with your customers. A digital twin allows you to build complex systems that otherwise simply would be out of reach.

Chapter 2 – five steps

Introduction

Now for the big question: how do you construct a Digital Twin of your system? On the basis, it comes down to five steps. Of course, there is more to it, and it will most probably take you several trials and tests to get the details right and familiarize yourself with all the intricacies, but these five stages will give structure to your workflow.

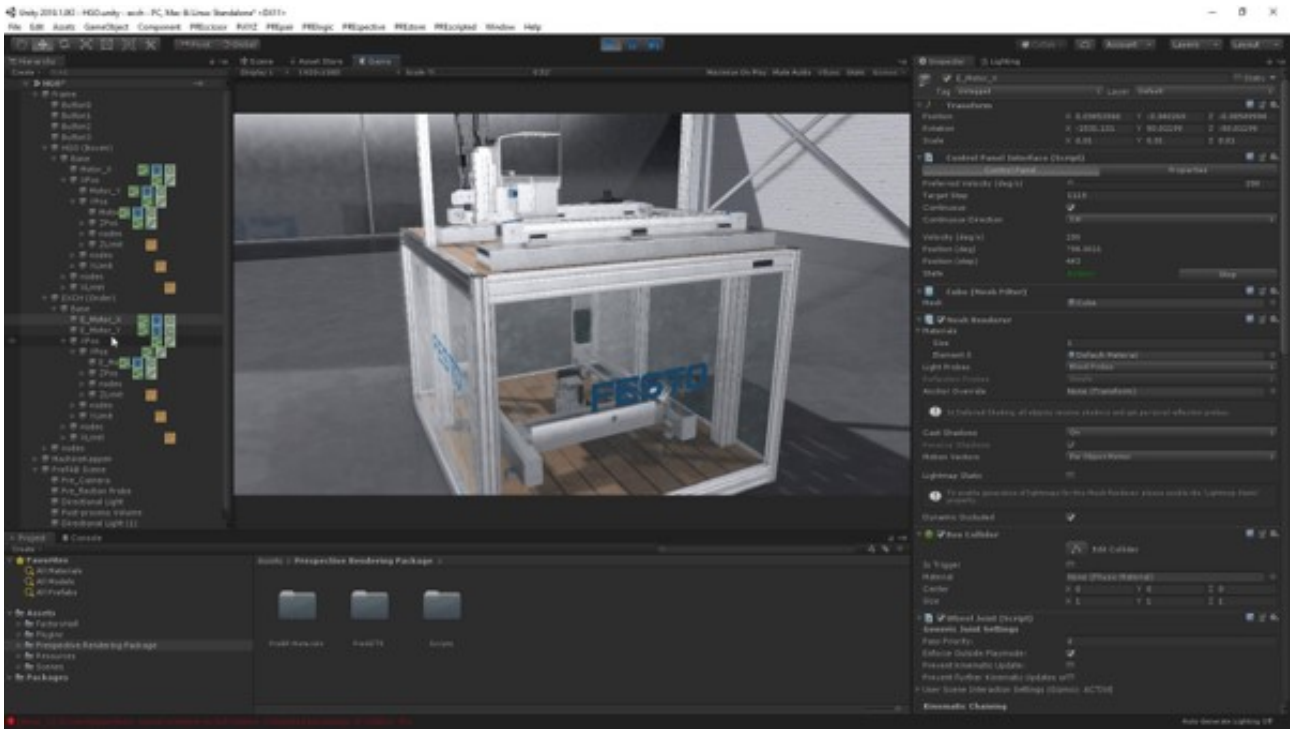
First of all, you import your 3D CAD file into Prespective. Then, you define all the components in that model. For all parts you have to specify whether it is static or dynamic, how it connects to other elements, and where it stands in the hierarchy of the system. The third step is to define the behavior of all components by describing their kinematic constraints and their mutual relations. When this is done, you want to connect your Digital Twin to the logic and control software. And last but not least, step 5 is to test and validate your system.

Note that you don't need to follow these steps to the letter, in the sense that you first do step 2 for all components and then move on to step 3. It might be more practical to run through the process, module by module. Whatever is most convenient for you.

Step 1 – Import CAD

To start building a Digital Twin, you first have to import your 3D CAD design in Prespective. Remember that this can be a rudimentary sketch, as the whole purpose of your virtual prototype is to start your visualization and simulations very early in the development process. What 3D Engineering tool you use to create your model – Autodesk, PTC, Siemens, Solidworks, to name a few – is entirely up to you. Prespective supports them all. Since Unity3D is an open platform, you can use an independent CAD importer like Pixyz to inject your model into Prespective.

Prespective allows for a little CAD engineering, but it is certainly not built to replace any of the suites mentioned before. 3D CAD tools are excellent for their specific task: drawing detailed vector-based models of your design. They are equally good at creating 2D images that form perfect input files for milling machines and the like. Prespective takes the virtual route and lets you do all this, and much more, in a digital environment with digital steel.



Optimize your design

Although the computing power of GPUs is growing every year, for high-resolution simulations it is wise to keep in mind that it is not without its limits. That means you have to edit and tweak your model before you can continue. A good starting point is to remove the parts of your design that are irrelevant for your simulation. That can be the structural parts such as plating, but also the screws and bolts. Of course, it depends on the reason why you are building your Digital Twin, but generally speaking, screws and bolts will have no real influence on the system performance in the virtual world. When the aim is to show perfect, high-resolution images, just for visual purposes, and the underlying physics does not take up much computing power, by all means, leave all the screws in.

Prespective offers a nice tool to dummy down your design: Prescissor. It allows you to filter out all screws, and other negligible components, by selecting one and letting the software search the entire model for similar parts. Afterward, in your 3D environment, you can easily hide or remove these trivial objects, thereby alleviating the load on the GPU.

Prescissor also lets you optimize the mesh of your model. In some parts, a lower resolution will suffice, while at the crucial core of the process, you want as many details as you can get. The tool makes it possible to differentiate between those areas, optimizing the model for its purpose. Furthermore, Prescissor can be used to change the scale of some parts in the model. This is sometimes necessary when you combine input from two sources.

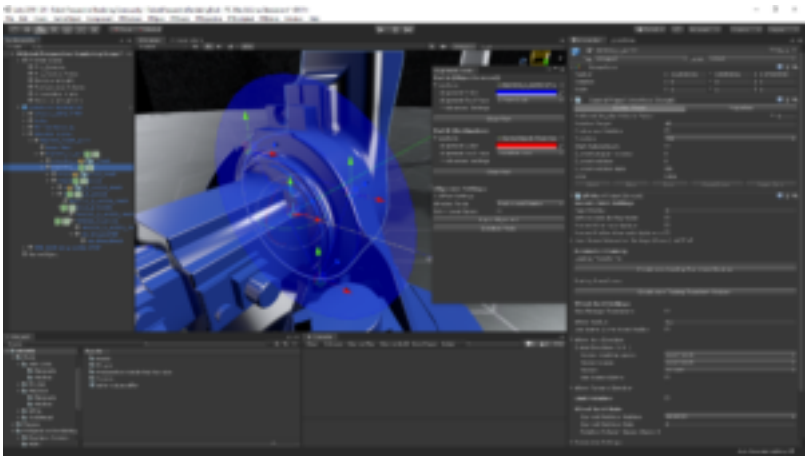
The final step in optimizing the geometry of your model is replacing objects with prefabs, which is very beneficial when you work with an updated model. You can simply drag and drop existing modules from earlier designs that previously were simulated, tested and commissioned.

Step 2 – Define components

When your 3D CAD model is imported and cleaned up in Prespective, it is time to let the platform know what is what. Define all mechatronic parts; the actuators, the motors, the conveyor belts, the switches and so on, and all sensors, light screens and other passive components. Also, define the static beams that are just in the design to support the structure and have no other function. Depending on the Digital Twin you want to build, they might be hidden later, hence enhancing the performance of your simulation. Remember that the more components you take into account, the heavier the strain on the GPU/CPU power will be.

Note that some static parts can still affect motions. For instance, there might be a pad to block or stop an object, or a rail to guide it along a certain path. These parts restrain the movement of other components and therefore, should not be removed.

Our tool offers an extensive library with many standard components, so labeling all the parts should not be too difficult. However, it can be quite a time-consuming task, depending on the complexity of your design.



Hierarchy

At this point, you want to have a detailed look at the hierarchy of your system. After all, the model is no random collection of objects: they are linked, they move, they collide, and they influence each other in many other ways. Prespective contains the Prepair module that provides you with all the tools you need to combine the parts of your CAD file into a workable and manageable setup for your application.

Prepair lets you cluster individual components that move as a block, to create groups that can be simulated together. Additionally, when combining input from different sources, it helps you to align objects, both translationally and rotationally, allowing you to straighten out any of the minor offsets you may encounter.

All these preparations and optimizations are imperative because it will greatly improve the performance of your visualizations and simulations. Save the GPU power for the core. Simulations can be measured in frames per second, just like in a video game. When you incorporate too much, and the frame rate drops below 24 fps, it will show in stuttering motions. Moreover, most physics simulations will break down below that threshold.

Case: Conveyor belt

To make things more tangible, let's consider a simple conveyor belt system. Up to this phase of the process, Prespective has no clue that some of the blocks and blobs in your model are actually meant to be a conveyor. In Step 2, it is up to you to explain this. Let the simulation tool know which parts of your model are interlinked to form this module. Label the belt, the drive roll – including the motor – and the passive roll. Also, tag the box on the belt as a moving object. These are all standard components in the Prespective library, so easily accessible in your drop-down menus.

By the way, the model of the drive motor in itself will normally contain several parts as well. Think of a casing, bearings and a rotor. You first want to group these elements into one cluster that you can then classify as being the driving motor for the belt. You also need to tell the software that the rotor is not a static object, but that it will be moving relative to the casing and the bearings.

Let's – for educational purposes – also include a sensor screen in the belt system that detects when the box is passing. Again you have to define which, until now, unidentified parts make up this component. What block is the emitter, where is the mirror situated and which part is the receiver? You now have defined a simple motion system, including the feedback loop.

Step 3 – Define the behavior

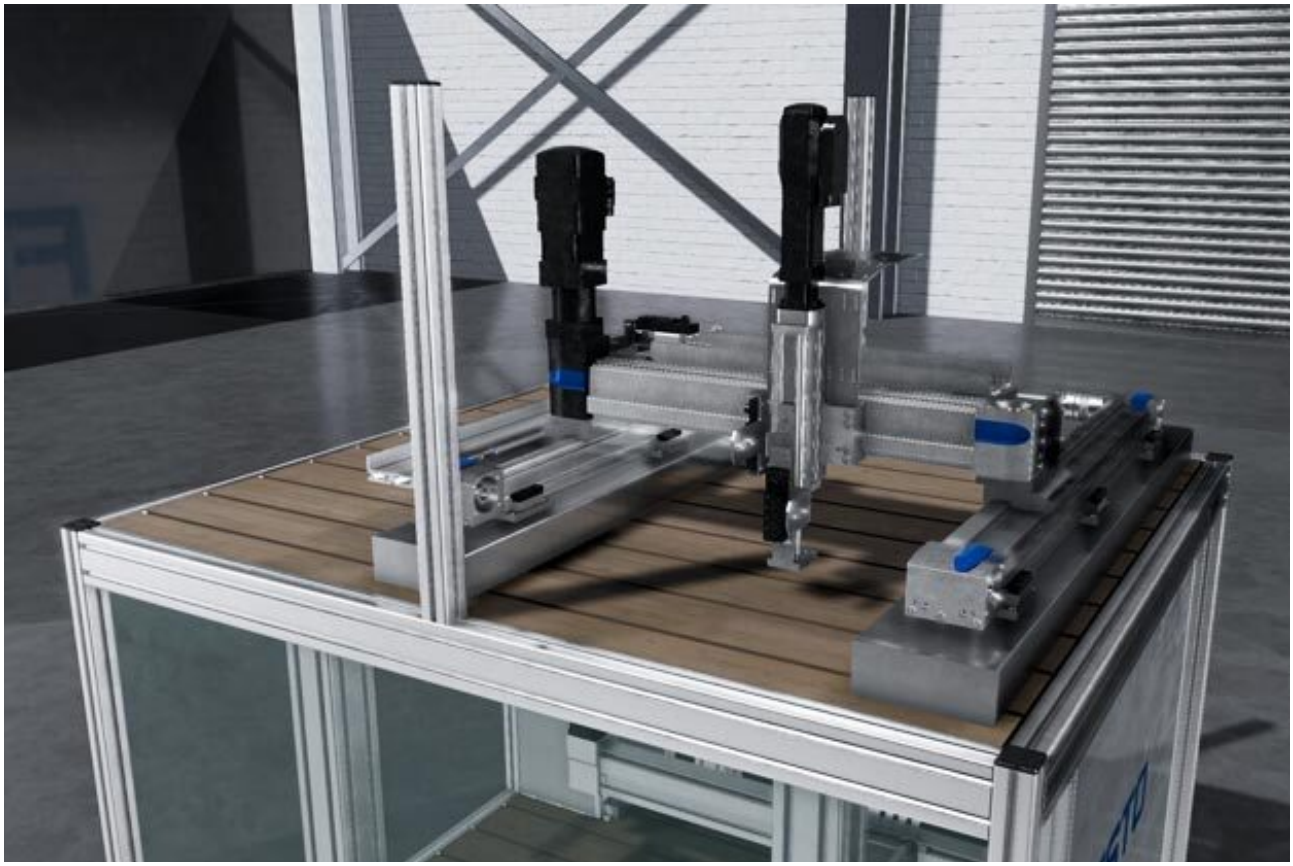
Next up is the assignment of behavior to the components of your Digital Twin. Many objects of the design will not be static but are moving relative to each other. You will need to explain these movements. Is it a linear or rotational motion, and what are the speed profiles?

Also, you need to define the input and output variables. A lot of these parameters will probably be predefined since many components are already known in Prespective. We propagate openness, so you can change all these settings within Prespective, if necessary, giving you the freedom to determine and finetune your own motion profiles. This is a feature that advanced users highly appreciate since it gives them the chance to look under the hood and tweak the settings according to their taste, how ever deep they want to go.

Some parts and components within your model will be highly connected, like, for instance, the separate joints of a robot arm. If one part moves, all the others will follow inevitably and according to a fixed set of motions. Your Digital Twin is unaware of this kinematic chain, even if you have grouped them in Step 2. So you need to link those objects and define their corresponding motions. Prepair can assist you in this task.

When considering and defining motions, pay special attention to areas where collisions can occur. Set limits, constraints and boundaries, or assign a denser mesh to this region, but don't overdo it. In a complex module, there will be thousands of objects. When you constantly try to compute if any of these are colliding, you will soon drain out the available GPU/CPU capacity. So you have to make

smart decisions about what you really want to know.



Case: conveyor belt

Returning to the example conveyor belt from Step 2, in Step 3, you connect the motor to the drive roll and the roll to the belt and passive roll. Then you define how the kinematic chain is composed. So the model knows that when the motor is turned on, the drive roll will start turning, the belt will take off, and the box on the belt will start moving as well. You also need to set the motion profile of the motor: how does it ramp-up, and how does it decelerate and stop?

The movement of the box requires special consideration. Its linear translation is directly linked to the rotation of the drive roll, in a one-to-one correlation. You can even describe it accurately in one dimension. For these kinds of movements, you may want to use a spline, since that makes the calculations for the simulation much easier. By restricting the movement to a spline, you decrease the possible paths, thereby minimizing the calculations it takes to generate a high-resolution simulation. You can quickly set up a spline in Prespective with easily accessible tools. All motions you limit with a spline will save you a lot of simulation effort later on in the process.

For the sensor screen in the belt system, you describe what type it is, for instance, whether it is an infrared or a laser beam sensor. And you should draw out the path of the light beam, from emitter to reflector to receiver. Also, you want to clarify what the output of the sensor will be: is there a box or not?

Physics

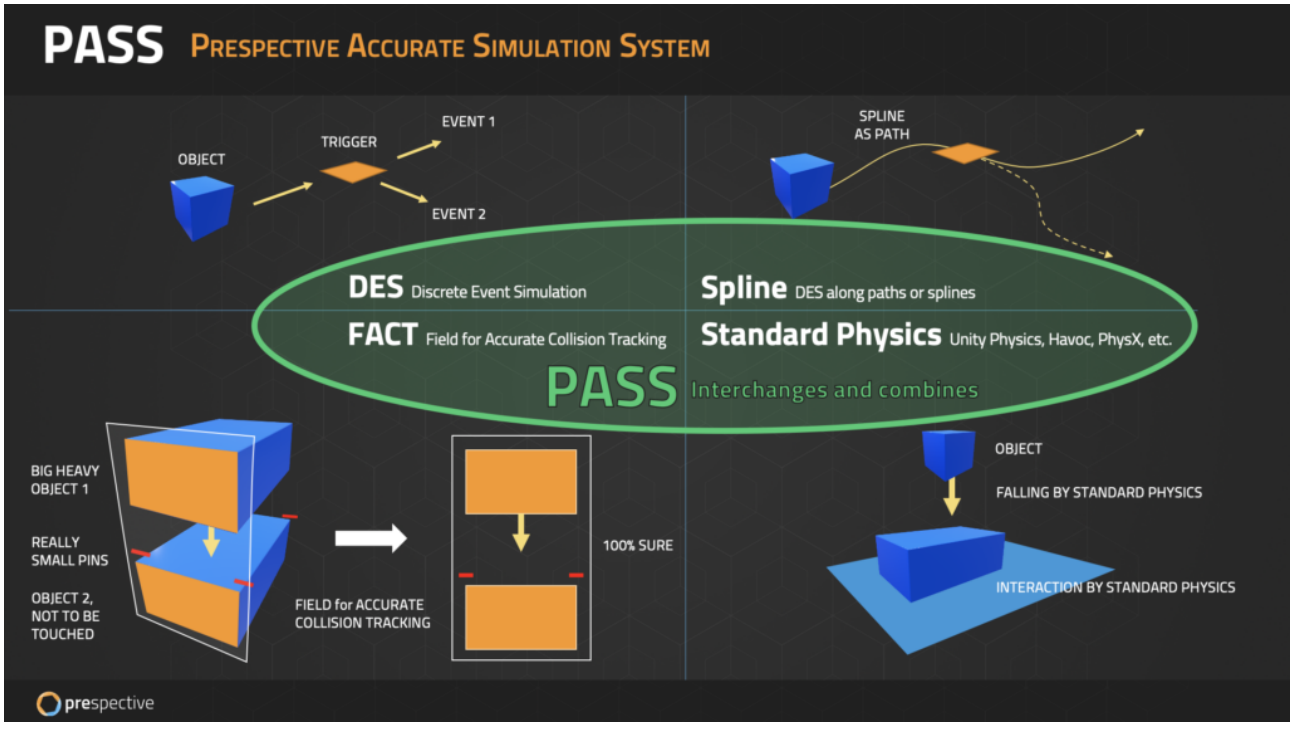
Another big factor in Step 3, is that you need to consider physics. In case of the box on the conveyor, obviously, it wouldn't make sense that it would fall through the belt as soon as you start the simulation. This is, of course, a very simple physics problem, that can be easily solved by the Unity3D physics engine upon which Prespective is based. But since real-life simulations are the cornerstone of a Digital Twin, it is crucial to get your physics straight.

Unity3D is constantly improving and getting more comprehensive with every release. Because we

highly value openness, you can also pick and choose from third-party physics engines such as PhysX or Havok (both coming from the gaming industry), or XDE (developed by the French research institute CEA). You can incorporate those engines without any problems. Whatever works best for you and your application.
Just like with the 3D CAD capabilities of Prespective, the power of Unity3D – and all other physics engines for that matter – has its boundaries. Some processes and situations, such as cutting-edge fluid dynamics or the intricate physics in wafer steppers, are just too complex and too specific. Software suites from specialized companies like Ansys, COMSOL, MathWorks or Wolfram are far better equipped for these kind of jobs. Again, because we propagate openness, you have the opportunity to work out the complex physics in those commercial tools – or even the proprietary software of your company – and import the relevant results to your Digital Twin in Prespective.

PASS

Prespective offers its own physics engine as well. It is called Prespective Accurate Simulation System (PASS) and is specifically targeted at industrial mechatronic systems. PASS consists of two segments: the Discrete Events Simulation tool and the 2.5D Physics Material Handling System.
DES is based on the previously mentioned splines. The tool will help you to set up the material flow in your Digital Twin quickly. Using the constrained paths of splines makes it easier to simulate all motions. Bear in mind that these splines do not have to be straight lines, but can be circular or any other shape, as long as it is a well-defined path.
Alongside these paths, many events can take place, being it a gripper, a push to change the direction of an object, or a sensor that registers when that object has passed. In the DES tool, you can program and keep track of all these triggers, creating almost a flow chart of the material flow.



2.5D Physics Material Handling System

The second tool within PASS is our 2.5D Physics Material Handling System. We developed this to overcome a shortcoming of Unity3D and other physics engines that are targeting the gaming industry. Those platforms are very good at tracking a larger number of objects, which is great in gaming since you are capable of generating and displaying, for instance, a large army with individual soldiers. The drawback of this feature is that it has a severe negative impact on the accuracy of all the objects. No big deal for gaming, maybe, but a potential showstopper for

industrial applications. In some areas in your model, you may need – and probably will need – a very high resolution, the highest you can get. With the 2.5D Physics Material Handling System, you can do precisely that. It allows you to tag regions in your scene where you want the highest resolution in your physics simulation.
Another feature of the 2.5D Physics Material Handling System will simplify the math problem tremendously so that you can reach better simulation results. To illustrate, let us assume you want to reproduce bumping golf balls in a moving box accurately. Very challenging in 3D, as you can imagine. However, by making a smart cross-section, you can bring it back to a much simpler 2D problem. 3D balls are reduced to 2D circles, which makes the calculations a hell of a lot easier. Moreover, since the golf balls are no longer built up out of mesh triangles but are real circles, you get the added bonus that your simulation will be more precise and accurate.
You can, of course, create these smart cross-sections on different, parallel levels when that makes sense for your application, or even in a plane perpendicular to the first. All of them will correlate with each other, but they do not make up a complete 3D image. Two perspectives will give you some idea of the third dimension, but not all information is present, as any machinist will tell you. The 2.5D Physics Material Handling System allows you to play with the number of objects in your simulation, and their resolution, so you can find your sweet spot.

FMU/FMI and AutomationML
When you want to optimize your workflow even further, FMU/FMI (Functional Mockup Unit/Functional Mockup Interface) or AutomationML might be interesting for you. These neutral data exchange formats make it possible to skip the first three steps of your Digital Twin development almost entirely. Files in these formats contain more than just the geometry. They also define the relations between objects and their behavior. For instance, when you import an FMU file of a robot arm in Prespective, the system will immediately understand the complete geometry, the kinematic relations and the potential ways of communication. Nowadays, more and more technology suppliers provide FMU or AutomationML files of their products to their clients, giving you the opportunity to streamline your design process.

Step 4 – Connect to logic

Up till now, you have been constructing a perfect Digital Twin. However, nothing is moving yet. The next step is to connect your model to logic. Every component that needs input or generates certain output has to be managed and controlled. You do this by mapping these components to some kind of logic. This can be the control software that the software department is developing in parallel. Or you can hook up a motor through our Logic Simulator tool to, for instance, a TwinCAT PLC. This way, your model will register a PLC message and knows what to do when a boolean is flipped to turn on a motor. The Logic Simulator will receive these instructions and send them through to the right component within Prespective, so the virtual motor will actually start to run.
Prespective makes sure to handle all instructions very precisely. Many database structures use the principle of “fire and forget”; messages are sent out without a double-check whether or not they are received and processed. The consequence could be that the simulation is not running accurately. The Logic Simulator confirms all messages and reports back that appropriate action is taken.
The generic set up of the Logic Simulator opens the door for proprietary or exotic PLCs or other embedded systems. For now, Prespective has standard communication settings for TwinCAT/ADS, OPC UA, MQTT and ActiveMQ.

Case: Conveyor belt
For our example, this step is quite easy, since there is only one motor and one light screen. What you need to do is connect the motor of the belt drive to your control software. From this point on, the virtual motor will know what to do when it receives an 'on' or 'off' message, or when the logic signals its input on the speed and direction.
Connecting the light screen to logic will ensure that when the box passes through the gate, a

message is sent to the controller. The software can then decide to stop the belt by sending that message to the drive motor.

Step 5 – Test & validate

Congratulations, all the hard work is done. Now the fun begins. You just have to push 'play' and see your machine in action. Virtually, of course. If you did all the previous steps flawlessly, this is your moment of joy. However, you have likely missed something, and your model does not show the right behavior. Maybe you just messed up a simple setting and your motor turns in the wrong direction. Whatever the reason, this is the time to debug, tweak and fine-tune your model.
Also, you will probably want to test some scenarios. After all, this is why you build your Digital Twin in the first place. Start with your Happy Flow, the simplest sequence of actions the system has to perform. After that, you can start experimenting with what happens in more unusual circumstances, or you can try out different settings and options to see what works best for your application.
To automate this test process, an experienced Unity user will be able to create a logger with a couple of use cases you want to try out. This way, you can easily run several scenarios and get a report back on what went right and wrong.
Now the true benefits of digital twinning finally present themselves. You can show your design to all stakeholders, check if it meets all requirements, discuss potential improvements and iterate the design to reach its optimum.

Case: Conveyor belt

When you push the play-button in Prespective, does the motor start running? In the right direction? Is the box moving as well? And does the belt stop when the box passes the light screen?
The example is too simple to test all kinds of scenarios, but you can probably imagine a million things that you would want to try out on your virtual prototype.

Chapter 3 – Gains

Extend possibilities

Why should you start with digital twinning? Here are the three main reasons. First of all, without the real-time visualization and advanced simulation capabilities, there is simply no other way to design sophisticated and ever more complex systems. For instance, there is no top-down approach in developing an automated guided vehicle or a self-driving car. You need simulation tools to check your logic and maybe even train your software.

Waymo did exactly that. The company started with the world model of the popular game Grand Theft Auto and upgraded that to a full-blown testing environment for its self-driving cars. In that Digital Twin, Waymo cars have driven millions of miles. The company uses Machine Learning to optimize the system behavior, but also to look for the best location and the ideal number of lidars, cameras and other sensors.

The best thing is that Waymo can try out the configurations that the Machine Learning algorithms suggest in its virtual world, let them run a thousand times, pick the best results and start a new iteration. Of course, it will have to validate the final design in real life, but the development process has been sped up tremendously.

Cost savings

The second argument for developing a Digital Twin of your system, is the total cost of ownership. The biggest saving comes from the fact that you can test, debug and optimize your design in a virtual environment. The iterative and agile approach helps to catch a lot of faults long before the milling machines start running, thus bypassing the need for many physical prototypes.

Note that it is mathematically impossible to construct a fully accurate world model – not in the least because of the observer effect – so you will not filter out all errors. Still, the industry is working hard to get the percentage as close to 100 percent as possible. At the moment, and based on real-life cases and experiences, on average, eight out of ten flaws are solved by applying Digital Twin technology. And it is common knowledge that avoidance of design errors in the early phases of the developmental process allows you to cut costs down the road, and can potentially even prevent total failures.

Acceleration

And finally, the third reason is time reduction. Two factors play an important role here: parallelizing and much better communication. To illustrate the first point, take a look at the work the software engineers are doing now. When their colleagues from mechanical engineering and mechatronics are busily developing the newest system, they have to sit on their hands until the first prototype is ready. And when they finally can start coding, then they are confronted by all kinds of accumulated misconceptions that have to be solved with digital duct tape.

With a virtual prototyping platform like Prespective, no one will ever be late to the party. As soon as there is a rough 3D sketch on the table, every discipline can set up the first iteration, which will grow gradually to the end product.

Secondly, a Digital Twin will greatly simplify the communication between all engineering disciplines and work truly multidisciplinary. Software designers can question mechanical engineers in more detail – and vice versa – because a virtual prototype brings the design to life. It will show everyone involved in real-time and in 3D how the system is constructed and how the different components are moving relative to each other. This is one of the most powerful assets of digital twinning, since it also allows you to easily evaluate your progress and ideas with (end) clients and iron out any unintentional misconceptions. You can also involve sales and prepare operators and maintenance engineers long before the final product is ready to be shipped.

Appendices

The Power of the Game Engine

The old-fashioned product development flow, still used by many companies today, is approaching the end of its life. It is no longer acceptable to first design the product in CAD, then build the physical prototype and finally call in the software engineers to make the system run. To avoid expensive redesigns and shorten the critical time to market, virtual prototyping is the way to go. This can be done with the Prespective platform, based on the Unity3D game engine.

Yes, gaming technology can seriously help you with your high-end industrial design. In fact, you cannot do without the real-time simulation capabilities a game engine has to offer. Many tools on the market claim to have that power, but since they are not targeted at real-time simulation, they generally fall short when you try to use them for virtual prototyping. Unity Technologies designed its the Unity3D game engine specifically to build virtual worlds, including real physics and true interactions between objects. It gives game developers the right platform to construct all the imaginary worlds they can think of, but with real-life graphics and natural behavior.

In 2005, Unity Technologies released its free cross-platform game engine, which has become one of the biggest in the world. Nowadays, more than half of all mobile games are built on this platform. But the capabilities of Unity3D are not only useful in game development. The industry also benefits greatly from this technology as an operating system for virtual worlds. For instance, with the game engine, it is very easy to change the color, or other options, of your car and immediately see how it would look. You can even extend the visualization to virtual reality and interact with your design. And you can do this in real-time, which is unique since no other software solution has the potential to generate 60 images per second. Game engines have that capability because they run on the computer's GPU instead of the CPU. And since GPU's can parallelize the calculations, they reach a much higher speed.

Supported by our partner Unity, we took their engine from the gaming world and transferred it to the industry. We now offer Prespective as a simulation and visualization tool for industrial applications. It runs on Unity3D as Excel runs on Windows, and lets you build virtual steel in real-time 3D. That Digital Twin allows you to verify and validate your design long before the hardware is built, making the development process faster, cheaper, safer and more reliable.

The Unity3D game engine is, in fact, so powerful that it is no longer meaningful to use a specialized visualization tool – which is a massive time saver. But virtual prototyping goes beyond visualization. On the level of physics, we are improving continuously. For games, this is of minor importance, but industrial clients set the bar high. Again, with strong support from Unity, the core of our engineering effort is to enhance the accuracy of real-time simulations, to such a degree that we can help you design even the most high-end systems.

++++++++++++++++++++++++++++++