– Metamorphers –
A Formalization of Transitions in Illustrative Molecular Visualization

A, B, and C

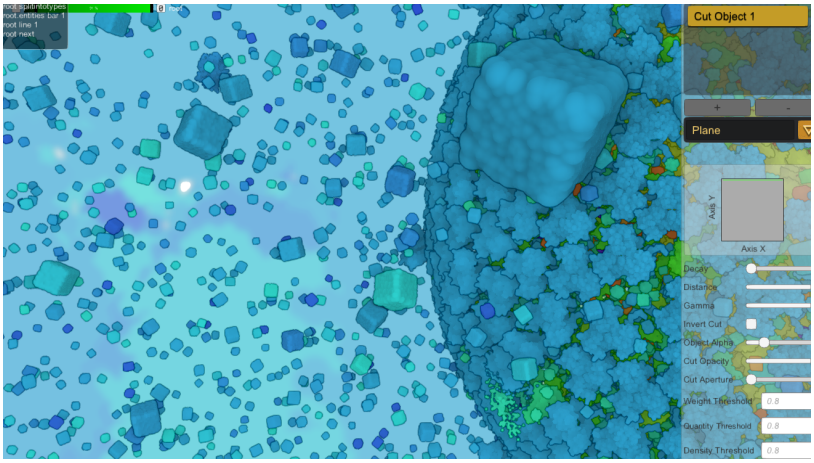

Fig. 1. tease me, then please me

Abstract— Illustrators in molecular biology often use animated transitions as a means to communicate complex phenomena. While these animations are popular, they still have to be created manually in 3D modelling software. We therefore provide illustrators with a more streamlined way for creating such transitions between various representations of molecular data.
We propose a formalization of the process of specifying these data representations in the form of a conceptual pipeline as our primary contribution. The pipeline consists of eight stages that can be grouped by their high-level functionality: target state representation, transition definition, and story helpers. As a result, data representations are specified in such a way that they form a continuous space. This space can be arbitrarily sampled to create a transition between the representations, e.g., in the form of animation or small multiples.
Our secondary contribution is a demonstration of our pipeline based on a proof of concept implementation. We created multiple operators for each pipeline stage that can be combined to achieve different types of transitions for arbitrary molecular data sets. One of these transitions that we designed in collaborations with illustrators, is a novel illustrative rendering approach for automatic schematization of mesoscopic data. In a qualitative discussion of our results, we received positive feedback from domain experts in illustration.

Index Terms—Molecular Visualization, Illustrative Rendering, Animation.

1 Introduction

The traditional approach for the visual communication of scientific insights is handled by illustrators and animators. Their goal is to attract the interest of broad audiences, and to convey complex phenomena in an accessible way. In recent years, we have seen an increase in the communication of topics from molecular biology **cite[drew berry, gj, david bolinsky, Gal McGill, janett iwasa]**, such as the splitting of DNA, the transport of oxygen in the the blood stream, or the comparison of molecular structures within a cell.

Illustrators have access to different narrative elements that can be combined with each other in order to convey their stories. The element of temporal sequence is used to convey changes of data states over time. The element of visual transformation creates different representations of the data in order to convey otherwise hidden aspects.

Visual transformation is necessary, because the molecular data data is complex. [explain complex data here or within the following examples?]Illustrators employ occlusion handling techniques, such as exploded views, to reveal the inner structures of a dense cellular data set. They employ focus & context techniques, such as the simplification of unimportant details of millions of atoms, in order to enable the presentation and comparison of cellular structures. While these two examples correspond to a transformation within the same visualization space, the depiction of abstract quantities or relationships, e.g., by bar charts or line graphs, correspond to the transformation to a different visualization space altogether.

For a successful knowledge transfer, viewers have to understand the relations between the original and the transformed data. How hard these relations are to grasp depends on how strongly an illustrative representation differs from the original data, how familiar the audience is with the presented data, and how complex the original data is (the number of objects that need to be related to each other, the amount of occlusion). It is therefore helpful for the user's understanding, to convey the transition, or *metamorphosis*, from one representation state to the other.

Animations are a powerful tool for conveying these metamor-

- *Author A is with Institute X. E-mail: a@x.com.*
- *Author B is with Institute Y. E-mail: b@Y.com.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.
Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

phoses, since they are capable of displaying them in a continuous way. The continuous presentation supports an intuitive understanding of the relationship between representation states. The presentation is self-explanatory and enjoyable to audiences [11]. Still, animations are also critically regarded by the visualization community. Tversky describes the fleetingness of the displayed information and the potential sensory overload and distracting nature of badly designed animations.

Alternative solutions for conveying transitions alleviate the fleetingness of the displayed information by sacrificing the continuous aspect of the representation. Examples are narrative sequences of small multiples or static images that supplement the missing information of the continuity with glyphs. While solving the problem of fleetingness, they have to be well designed to be understandable.

These approaches are most suited for circumstances where little to no interaction is possible or desired, such as in presentation scenarios. In scenarios where interaction is possible or desired, other approaches, such as coordinated multiple view (CMV) visualizations can be used.

However, no matter which medium is used for the illustrative depiction of these stories, these transformed representations and the transitions to them have to be modeled manually by the illustrators which is a laborious and time-consuming task. The authors build their visualizations with complex molecular models, e.g., of a cell, consisting of millions of atoms that form tens of thousands of molecules, that are loaded into the scene of a 3D modeling tool. Considering that new discoveries are made frequently in molecular biology, these representations and their transitions have to be re-modeled when new insights are found, which makes the effort for maintaining them even higher.

We therefore want to assist illustrators by presenting our primary contribution: a formal description of a uniform method for creating target representations for spatial biological data sets, as well as the continuous transitions to these representations - describing the complete metamorphosis of the data representation, so to say.

The output of this method is a continuous space between two representation states. This space can be arbitrarily sampled to create a transition between these representations, e.g., continuously, in the form of an animation, or sparsely, in the form of small multiples.

The method is described by a pipeline that consists of eight stages that can be grouped by their high-level functionality: target state representation, transition definition, and story helpers. Each stage supports operators, so called metamorphers, that are responsible for one aspect of the metamorphosis.

As our secondary contribution, we demonstrate a proof-of-concept implementation of this pipeline based on three use cases that we developed in collaboration with illustrators. One of these results, a novel illustrative rendering approach for automatic schematization of mesoscopic data, is our third and final contribution.

2 Related Work

Animation and transitions between visual and data representations have been employed in many contexts. They serve as powerful tools for the dissemination of complex relations in space, time and abstract dimensions. They are frequently used in visual story telling and for the depiction of correspondences between data representations as well as visual representations.

2.1 Visual story telling

Kosara and Mackinlay [7] note that for a long time the focus of visualization research was on exploration and analysis but that presentation of findings especially using elements of story telling should be a research focus of equal importance. They list prominent examples of story telling that include animated transitions for trend analysis and dissemination.

Robertson et al. [11] have looked at trend visualizations in depth. Three methods including animated transitions were analyzed. They found that animated transitions of data was reported to be more enjoyable and exciting to users. They also found that it was significantly faster when used for presentation but less exact and less effective for analyzing data. These results encouraged us to use animated transitions to facilitate presentation of complex molecular data.

Segel and Heer [12] analyze story telling with narrative visualization and point out different approaches. They distinguish between author- and reader-driven elements in a narrative and speculate that "exploration of transitions between them presents an exciting area for researchers and practitioners". Along these lines Wohlfart and Hauser [15] presented a guided interactive volume visualization approach. Their system enables authoring and editing of visualization stories that give the user partial control over the exploration.

Ma et al. [10] describe the need of new scientific visualization methods that can be used for scientific story telling. Among other issues they suggest that novel transitions between different coordinate systems are needed to address the overwhelming complexity involved in today's scientific story telling.

2.2 Transitions between visual representations

Kosara et al. [8] use 3D scatter plot matrices to establish a link between physical layout and the abstract dimensions of the data. Their work is an early attempt to connect information visualization and scientific visualization using points as data primitives.

Guilmaine et al. [3] compare different animated transitions of tree structures. They find that hierarchical animation is better suited for tracking of changes. Basch [1] describes possibilities for animated transitions between volumetric rendering and abstract views like histograms and scatter plots. Heer and Robertson [4] show animated transitions between different visual representations of statistical data. These works make use of staging and staggering which are techniques that reduces occlusions and clutter. Our approach is based on these findings and formalizes animated transitions between spatial and abstract attributes.

Huron et al. [5] present the visual sedimentation metaphor for the animation of data streams. This metaphor allows to transition from discrete visual elements to a continuous representation. We employ a similar metaphor for the transition of spatial objects to abstract charts.

Exploded views [2, 9, 6] have been used in different contexts as a technique for illustrating spatial or hierarchical relations of parts of an object. They are frequently used in animations or interactive systems and reported to be effective in the dissemination of complex layouts. Our approach is similar to exploded views, since it animates the decomposition of the object into its parts. However, the spatial displacement is not only driven by the visibility of the individual parts, but also by additional data attributes.

2.3 Alternatives to animated transitions

Small multiples [13], coordinated multiple views, and static images in conjunction with annotations, traces and glyphs are alternatives that are typically used to visualize transitions, trends, correspondences or sequences [11]. Tversky et al. [14] summarize cognitive studies on the benefits of animation. Although, they conclude that animation alone has not been convincingly demonstrated to be superior to static illustrations, other findings of their study suggest a direction for research on animations in visualization. They report that animation together with basic interaction methods like pausing, partial re-playing, zooming and change of perspective might be the key to enhance the effectiveness of animation. Further, they acknowledge that the alternatives to animation (such as sequences of static illustration) are surprisingly hard to design and are therefore not an easy target for computer automation.

3 The Pipeline

3.1 Overview

In this section, we present the eight stages that our pipeline consists of. The pipeline operates on the following data format.

- Instances: instances have a spatial position, orientation, and geometrical representation, and can have an attribute vector. in our case, instances are molecules of different types that are composed of atoms. these atoms are represented by points and are the primitives in our data.

- Subsets: subsets are grouped instances, in our case groups of molecules. groups can be formed according to any internal or external attributes of instances, such as type or shape similarity.

- Entities: both instances and subsets are the entities that our pipeline operates on. this means, a pipeline operator accepts either individual instances and/or groups of instances, e.g., in the layout stage, when repositioning a single molecule or a group of molecules.

todo: where to describe cellpack format?

Our pipeline takes a molecular data set in the described format and produces a user specified number of frames that show a transition from the provided original representation to a specified target representation.

The eight stages of our pipeline can be grouped into two high level tasks: the first four stages are responsible for creating a target representation. The final four stages are used to define the transition from the original representation to the target representation. Each stage supports certain types of operators. These operators determine how a stage proceeds with the supplied data. Operators have to be implemented for each stage in consideration of desired target output. Here is where design considerations come into play. To convey their message, illustrators apply a combination of the three high level tasks of occlusion handling, and transformations within or across different visualization spaces. Subsequently, most pipeline stages supply measures for fulfilling these tasks with their individual means. These can, in combination, then enable other lower level tasks, such as, comparison,.. **[check brehmer task taxonomy]**

In order to achieve a desired result, the respective operators have to be selected and parametrized for each stage. Depending on what kind of transition should be achieved, not all stages are mandatory. Further, the pipeline can be repeated for more complex stories or multi stage transitions. In the following we will discuss each stage in terms of its function within the pipeline and in terms of how it operates on the supplied data. How these stages can be applied in practice, will be demonstrated in the next Section, where we will discuss the creation of three exemplary transitions based on our proof of concept implementation.

3.2 Target State

The visual appearance of the final representation is defined in these stages.

3.2.1 Data Transformation

Before they start to visually transform their data, illustrators have to decide, which parts of the data and in which granularity it is handled, e.g., to determine, which parts to explode, or which parts to merge and render in a simplified manner. As such, the components on (most of) the following pipeline stages are operating on individual subsets of the data. Data Transformation, as the first stage in our pipeline, is therefore concerned with creating these subsets - or more precisely, a hierarchical scene-graph of subsets where the leave nodes are the individual molecules. These subsets are the building blocks that the final representation is constructed from.

Operators in this stage therefore define, according to which criteria to create these subsets. For instance, one operator could create a subset that contains all molecules of the same type, another according to the molecules' spatial positioning.

On a technical level, this means that operators of this stage will receive a list of entities (molecules or subsets) as input - will then group these entities according to specified criteria - and create a number of subsets, e.g, as many as there are molecule types. Each entity that fits the respective subset criteria is assigned to it as its child. Each child entity stores the id of its designated parent. The output of an operator will therefore be a list of the newly created subsets with the respective child / parent assignments.

To create a scenegraph, the operators of this stage have to be called hierarchically on the data. A user could, for instance, at first partition the scene spatially and then partition the resulting subsets by molecule type.

Multiple time steps, different data sets, or simply multiple copies of the same data can also be represented as high level nodes within such a scenegraph - provided the necessary operators to create or load them are implemented.

3.2.2 Annotations

This stage allows the illustrator to add various forms of annotations to the scene. Illustrators can use these annotations for multiple purposes: simple text labels could assist story telling by describing what is currently happening in the scene. Icons that point to specific events can be used to steer the viewer's attention. These elements can also be combined to create frames of reference, such as coordinate axes, scales, and legends, that can improve the viewers' orientation within and their understanding of the presented information.

Operators of this stage should therefore, in the simplest case, accept text input and should be able to create or load icons and other graphical representations. Annotations can be independently positioned and parametrized, but they can also be logically attached to specific entities, e.g., for creating a relative position or extracting and displaying an entity's attributes.

[we could also move this behind layouting]

3.2.3 Visual Transformation

Operators in this stage have access to the visual variables of transparency, color, and shape. Illustrators use these operators to change the visual mapping of parts or all of their data, in order to achieve a transformation within the same or across visualization spaces, e.g., to support tasks like simplification, abstraction, and occlusion handling. Operations can be applied on any level of the scenegraph, i.e., on individual molecules or subsets of molecules.

This allows the transparent rendering of occluding molecules, or the morphing of entities to different shapes, as can be seen in [Figure 2 - graham video].

From a technical perspective, the operators in this stage can implement specific shaders for creating a desired effect, e.g., by changing the molecule geometry, or doing post processing on the rendered image of a subset in a separate rendering pass.

3.2.4 Spatial Transformation (Layouting)

While the previous stage handled the visual variables of transparency, color, and shape, this stage is concerned with the transformation of the spatial variables of position, rotation, and scale. As the final stage of this part of the pipeline, it completes the transformation of the data, i.e., the creation of the final target representation.

With the spatial re-arrangement of the molecular data, various results can be achieved: occlusion handling, e.g., by exploding the dense data to reveal the inner workings of a cell. The noisy distribution of different molecules can be re-arranged, to simplify the complex and dense data so that it better fits the viewer's mental model. Elements could be positioned so that they represent an altogether different visualization space, such as a network that shows interactions between molecules.

The input for a layout operator is always a node in the scene hierarchy, i.e., an entity containing a list of molecules or subsets. For each child of the supplied node, the operator calculates a new position rotation, and scale - depending on what kind of layout the respective operator is implementing. The output of a layout operator is therefore a new position, rotation, and scale for each child of the supplied node. Each entity has a vector of control points that stores positions and rotations and is initialized with the entity's original position and rotation (in the case of subsets, the bounding box). The new values calculated by the layout operators are added to this vector.

Layouts can be stacked upon each other - meaning they are called multiple times for the same node - this creates additional control points that can be used to transition the data into intermediate representations before arriving at the final state. The final representation is determined by the last layout that has been called. A user could, for instance, first

call an explosion layout, to reveal the inner parts of the dense data set, before rearranging these parts into a simplified form.

3.3 Transition

These stages are responsible for defining the transition to the target representation, i.e., how each entity reaches its target position and when it reaches it's target form.

3.3.1 Camera Operators

Control over camera parameters allows illustrators to steer a viewer's attention towards specific events or details in the scene, in a guided navigation fashion. An operator could also steer the camera to follow the development of a transition, if it leaves the current view frustum, to assure visibility of the entire data.

Operators in this stage modify the camera position & look-at vector. As such, they create a list of control points for these two values. For guided navigation, an operator could access the positional control points of user defined entities, for instance. A timing and trajectory is defined in the respective stages for the interpolation between these control points.

3.3.2 Trajectory

This stage describes for each molecule, how it reaches the target position and the control points in-between that have been defined in the spatial transformation/layouting stage. This is in the simplest case a linear interpolation between a molecule's control points. However, if applied wisely, illustrators can use this stage for multiple purposes: Simplification: they can make the transition more readable and visually engaging, by bringing structure into its elements' motion paths, e.g., by creating a well positioned point in space for each molecule type through which it has to "travel" to reach its target position. This can also serve occlusion handling, e.g., by placing these transit points in a way that the molecules flying through them reveal essential parts of the scene. Trajectories can also encode topological information that was not present in the original data, e.g., in the form of meaningful data highways that transport molecules from one cellular compartment to another, or to simulate the flow of blood between two organs.

To achieve this spatially structured transition, operators in this stage can implement techniques such as edge bundling. This means that an operator can introduce new control points into the motion path of a molecule, e.g., to re-direct it so that it does not occlude the existing scene. Operators in this stage also determine, whether to interpolate linearly between control points, or whether to use other strategies, such as cubic splines.

As such, this stage, takes a subset from the scene hierarchy as input. The child elements of this subset are analyzed in terms of their control points (at the very least original and target position) and their relation to the control points of the remaining elements of this subset (or the entire scene). Depending on the operator type, new control points that fulfill a desired function, such as clutter reduction in the form of more orderly transition paths, are created as the output of the stage.

3.3.3 Timing

The timing stage is responsible for the temporal coordination of the transition by determining the order and speed at which entities or subsets reach their target position and visual form. As such, this stage is the illustrators' canvas in the temporal dimension.

While the inherent purpose of this stage is to encode chronological information, i.e., the sequence of events, illustrators can also use it for additional purposes: they can make the transition more readable & visually engaging by reducing the number of simultaneous visual stimuli, i.e., by starting the transition of indicisual molecules / subsets with a time offset to each other, thus creating staged transitions/animations [cite heer?]. Timing can also be used for occlusion handling, i.e., by first starting the transition of elements that occlude important structures or that are closer to the viewing position, while other elements remain in their positions until they are revealed. Further, timing can also be used for controlling visual stimuli. The transition can be timed in a way that the attention is steered to a currently important structure or event by animating what should be in focus while keeping the remaining scene still.

As such, operators of this stage, depending on the intent that they implement, create a time curve for each entity. This time curve determines the start time and speed of the spatial and visual transition. Optionally, also an end time can be created, i.e., to determine, when an element should cease to exist but also, when the transition should stop, i.e., whether an element should reach its final destination as well as how close it should get. This can be used to create a target representation that looks like it stopped in mid-motion.

3.3.4 Sampling

This final stage determines the sampling strategy, i.e., at which rate the transition is sampled. The operator, that the illustrator chooses, therefore determines, whether the transition will be continuous or if sparser sampling will be applied, e.g., to extract a narrative sequence like in a comic strip. The lowest rate would therefore just output the final image of the target representation.

While the previous stages of the pipeline can be called multiple times in order to create more complex animations/transitions/stories, the sampling stage is final, as it ties together the information from the previous stages to sample the visual result.

Before the sampling starts, the scene hierarchy is flattened again - which means that all information from higher level nodes is projected into the leaves, i.e., the individual molecules. The transformation matrices of higher level nodes are multiplied recursively with all their lower level children for each controlpoint - so that each molecule knows about its global position for each control point.

The sampling stage then uses the output of the timing stage to determine at which rate/speed to interpolate between control points for each molecule. And it uses the output of the trajectory stage to determine how to interpolate between the supplied control points, e.g., linear, cubic,..

The sampling stage can also determine if the transition is pre-baked or sampled on the fly in real-time. This makes it possible to use a continuous pre-baked transition to create, for instance, a single result image that encodes the missing transition information within glyphs. These glyphs can be calculated from the continuous pre-baked information.

The continuously sampled information could also be exploited to create a hybrid visualization that displays a discretely sampled narrative sequence of small multiples. Upon clicking one of these small multiples, it could show the continuous transition to the next discrete element in the narrative sequence.

4 Use Cases

In this section, we will demonstrate our proof of concept implementation based on three exemplary use cases that we developed with illustrators. We will first describe our data and then motivate these use cases, [actually, we already motivated them in the introduction..]

After this, we will motivate the components that all three use cases have in common. And finally, we will describe each use case's particular components and how their application produces the desired representations.

Our use cases are demonstrated on a data set representing an HIV particle engulfed in blood serum. The data consists of 42 types of molecules, that have a total of XXX number of instances, consisting of a total number of YYY atoms. The data can be partitioned into three closed compartments: the blood serum, the outer shell (membrane?) of the HIV particle, and the nucleus within the particle. They were supplied to us in the cellPACK format [cite cellpack].

4.1 Explosion of Molecular Structures

Since molecular structures are very densely packed, it is impossible to inspect the outer and inner structures, e.g., of a virus, at the same time, without applying some sort of occlusion handling technique, such as exploded views [cite some fundamental exploded view paper] [kind of have to repeat the argumentation from the introduction here - i guess

here i should be more specific..]. [why is it necessry (problem statement)][what does it preserve(focus)][what does it sacrifice]

4.2 Schematization of Molecular Structures

In scientific illustrations, atlas based views schematic representations of cellular structures are used to give an insight of the molecular and atomic composition of these cells. The focus lies therefore on displaying individual molecules to convey their atomic composition, as well as on selected constellations of molecules to convey the structures that they collectively create. However, the original data model consists of millions of mostly noisily distributed atoms. To achieve this schematic representation therefore, one the one side, specific molecules have to be chosen as representatives of their type to be displayed at a sufficiently large scale (focus), while suppressing the noisy distribution of millions of atoms (context). On the other side, molecules that do not have a noisy distribution but form coherent structures, such as cell membranes, have to be preserved. This representation sacrifices the low level detail of the noisy atomic distribution, i.e., the exact count, position and shape of individual molecules. It aims to preserve the shape and size of compartments as well as molecule type overview, compartment relation (zugehrigkeit), and molecule shape. This metamorphosis corresponds with a transformation within the same visualization space.

@discussion of results: transition = more intuitive to follow.

because this type of automatic illustrative schematization has not been presented/solved yet in literature, we present it as our tertiary contribution...

4.3 Representation of Quantitative Relations

For the explanation of the molecular data, often also the relation of quantitative aspects is important for its understanding, e.g, how many different molecules of each type does the HIV particle contain, or how much of the volume within the particle does each type occupy? Graham Johnson answered this question in his award-winning video demonstration of the decomposition of an XXX cell. In order to achieve this sort of representation, the molecular data has to be transformed into a different visualization space. A stronger abstraction than in the previous two examples takes place. To convey these quantitative information about aggregated volumetric and count information, shapes and positions of compartments and molecules are sacrificed. Since this transformation is across visualization spaces, the target representation is harder to relate to the original representation.

@discussion: here, large distance between representation forms in visualization space - a continuous transition is therefore especially suitable for conveying the transition..

4.4 Common Components

The following metamorphers are involved in all three usecases.

4.4.1 Grouping by Type Range

In all use cases, we want to partition the data into the afore mentioned three high level subsets of the blood serum, HIV membrane, and HIV capsid. This is necessary to be able to operate on each subset individually, e.g., when exploding these compartments separately [mention here already how we use it in each usecase?]. We therefore need subsets that contain entities of a specified type range. The metamorpher therefore creates as many subsets as there are ranges specified, and parses a given list of entities. Entities of a type matching a defined range, are then assigned as children to the respective subset.

4.4.2 Text Labels

4.4.3 Icons / Textures

4.4.4 Translate?

4.4.5 Rotate?

4.4.6 Linear Interpolation

4.4.7 Staging

4.4.8 Spatial Time Offset

4.4.9 Guided Navigation

4.4.10 Continuous Sampling

4.5 Specific Components

[will be placed in the respective usecase]

4.5.1 Grouping by Type - use case C

We need to visually or spatially transform all molecules of the same type in a uniform fashion, e.g., to spatially transform them into a collective shape. [refer to all usecases here?] The output of this metamorpher is therefore a list of subsets that contain entities of the same type. The metamorpher parses a given list of entities and creates a new subset for each new type-ID that it finds. Entities of the matching type are then assigned as children to this subset.

4.5.2 Cloning

4.5.3 Spatial Grouping

4.5.4 Blurring

4.5.5 Morphing

4.5.6 Bar Layout

4.5.7 Line Layout

4.5.8 Schematic (Scale & Rotate)

4.5.9 Transit Points / Bundling

4.5.10 SLERP

4.6 Explosion of Molecular Structures

4.6.1 Applied Metamorphers

–data transform: 1) [type range grouper] create three high lvl subsets: the blood serum, the membrane, the capsid 2) [plane splitter] spatially partiton each subset with the same plane (each one could also be split with a different plane theoretically)

–annotations: 3) text labels: ”serum”, ”capsid”, ”membrane”

–blending: no blending applied here

–layouting: for each high lvl subset: 4) [translate] upper part up, lower part down 5) [rotate] rotate upper part a bit up, lower part a bit down 6) (for all except the first subset)[translate] the subset to the left (double the amount of the previous subset’s BB width) 7) position label on top/below each high lvl subset

–trajectory: 8) [linear interpolation]

–timing: 9) introduce a [delay] for each consecutive layout operation (translate sideways of highlvl subsets & translate up/down for low lvl subsets) 10) [] let some random molecules not finish the transition 11) [] let labels appear after animation is finished

–navigation: 12) guided navi as subset explosion & transition use more and more screenspace

–sampling: 13) continuous sampling

4.7 Schematization of Molecular Structures

4.7.1 Applied Metamorphers

–data transform: 1) [type range grouper] create three high lvl subsets: the blood serum, the membrane, the capsid 2) [copy] the entire partitioned data set because we will render it in two different representations =¿ results in copy A & B

–annotations: do the circular annotations?

–visual transform: A: blurrin & increasin contrast of alpha channel for nice charp edges of each subset individually B: scalin

–layouting (spatial transform): A: renderin them on top of each other (occlusion management) B: rotatin

–navigation: not necessary, because we stay in the same visualization space / spatial extents

–trajectory: B: linear interpolation between rotations

–timing: staged per compartment distance field left to right for each

–sampling: continuous

4.8 Representation of Volumetric Relations

4.8.1 Applied Metamorphers

–data transform: 1) [type range grouper] create 3 hierarchical compartments by typerange 2) [single type grouper] split all three by type =¿ subsets containing all molecules of same type for each hierarchical compartment

–annotations: 3) [txt-label] for each type subset 10) [representative] molecule for type subset

–blending: 4) [slice-flatterer] create target rep for each molecule with blending: slice with hight representing the molecule volume

–layouting: 5) [bar layout] for each type compartment - the layout puts all molecule slices on top of each other to create histogram bars 6) the [line layout] puts all the bars within a compartment side by side 7) the [translate layout] then moves the new center for the next line layout for the next compartment

8) layout: put label below below each bar 9) layout: put rep below label

–trajectory: 10) [top-down-fall-in-component]: for each molecule in bar subset, we add a control point to the top of the bar, so that the molecule seems like falling into the bar instead of going straight from its original position to the designated position within the bar.

–timing: 11) [distance offset] a plane can be positioned in space. the distance of each molecule/compartment to this plane is measured and used for a the creation of a time offset that specified when each molecule is allowed to start its transition

12) timing: make label & rep appear when animation is finished

–guided navi: 13) [whole-data-in-view-keeper] update the camera pos & lookat to keep all molecules in the view frustum.

–sampling: 14) continuous sampling

4.9 PoC Pipeline

[contains specifics about our pipeline implementation] **[move to implementation section]** implemented in cellVIEW: give some details about cellVIEW): unity, supports loading &rendering of cellPACK data, renders molecules as point clouds

desc was jeder component beisteuert um die beiden reps zu erzielen

4.10 Metamorphers

[just for reference] [describes PoC components that we implemented per pipeline stage]
[should we describe beforehand what (which representations) we try to achieve with these implementations?]

4.10.1 Partitioning

spatial subselection by clipping plane/object,
splitting into types,
splitting into type ranges
cloning

—not showcased but implemented: *spatial splitting operations into equal parts (pie splitter),

4.10.2 Annotations

text labels, icons (textures), maybe: data representatives (for each type), optional addon to each operator: connectors between annotation & assigned subset

4.10.3 Blending / Morphing

blur into collective shape, morph,

—not showcased but implemented: blending to texture

—other not implemented & not used examples: unfolding

4.10.4 Layouting

bar, line, translate, rotate, scale, schematization layout = combi of scale & rotate but selection of elements where what op is applied is very specific

—not showcased but implemented: slice, sphere, circle

4.10.5 Camera / Guided Navi

[whole-data-in-view-keeper] update the camera pos & lookat to keep all molecules in the view frustum.

other examples:

4.10.6 Trajectory

linear: from A straight to B, bundling/transit points

—other not implemented & not used examples: curved

4.10.7 Timing

offset function: delay timing by distance to plane & staging by type ... anything else?

—other not implemented & not used examples: offset according to: inherent spatial parameters (position, distance to other entities, density)

4.10.8 Sampling

continuous sampling,

—other not implemented & not used examples: sparse (comic strip / narrative sequence),

post-processing operators: motion blur,

5 Results

in this Section, we discuss the outcome of our proof of concept implementations

5.1 Representation of Volumetric Relations

5.2 Schematization of Molecular Structures

things that should be preserved: depending on the usecase / representation - not all properties are essential for each representation form.

- instance level: position of instances
- type level: shape of types, relative size of types, color of types (?), number relation between types
- compartment level: shape/pos/size/color of compartments / structures
- global level: volumetric dimensions of the data
- todo: what else?

6 Discussion

6.1 additional outcomes

side products of our pipeline - free of charge / easily achievable with minimal additional effort: sequence: comic strips, static image with glyph

6.2 generalization

compatible data formats: point cloud, volume, polygonal?

compatible data types: dense hierarchical data with lots of repeating entities of a few dozen types

6.3 general discussion

tversky: congruence principle: the structure and content of an external representation should match the structure and content of the desired mental representation.

apprehension principle: the structure and content of the external representation should be readily and accurately perceived and comprehended.

6.4 occlusion

is of course a central factor that could hinder a user's understanding of a transition. can be dealt with at various stages of the pipeline. grouping: choose groups so that they dont interfere with each other (depending on how you plan to transform the data) TODO: come up with examples that describe which kind of grouping would be beneficial for which kind of target state e.g., group the data first into layers parallel to the viewing plane that basically peel the data layouts inner bar: within the bar occlusion does not matter, as long as the bar represents entities of the exact same type todo: other inner layouts & general outer general: the placement of entities/subests should be chosen so that they dont occlude each other from the viewing angle. maybe we should not distinguish between inner and outer layout since this kind of restricts us to having two layouts instead of n.

then again, we could argue that there can be n-outer layouts, and one inner layout morphing: same as inner layout: within the shape occlusion is not an issue

trajectory: the trajectories offer two of techniques for avoiding occlusion.

edge bundling: setting control points so that paths don't occlude each other todo: anything else?

timing: coordination of timing and speed of elements, e.g., staging

6.5 future work

implementation of comic strip / animation hybrid

7 Conclusion

Acknowledgments

The authors wish to thank A, B, C. This work was supported in part by a grant from XYZ.

References

[1] C. Basch. Animated transitions across multiple dimensions for volumetric data. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, oct 2011.

[2] S. Bruckner and M. E. Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, Sept. 2006.

[3] D. Guilmaine, C. Viau, and M. J. McGuffin. Hierarchically animated transitions in visualizations of tree structures. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 514–521, New York, NY, USA, 2012. ACM.

[4] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, Nov 2007.

[5] S. Huron, R. Vuillemot, and J.-D. Fekete. Visual Sedimentation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2446–2455, 2013.

[6] O. Karpenko, W. Li, N. Mitra, and M. Agrawala. Exploded view diagrams of mathematical surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1311–1318, Nov. 2010.

[7] R. Kosara and J. Mackinlay. Storytelling: The next step for visualization. *IEEE Computer*, 46:44–50, 2013.

[8] R. Kosara, G. N. Sahling, and H. Hauser. Linking scientific and information visualization with interactive 3d scatterplots. In *Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 133–140, 2004.

[9] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3d exploded view diagrams. *ACM Transaction on Graphics*, 27(3):101:1–101:7, 2008.

[10] K. L. Ma, I. Liao, J. Frazier, H. Hauser, and H. N. Kostis. Scientific storytelling using visualization. *IEEE Computer Graphics and Applications*, 32(1):12–19, Jan 2012.

[11] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, Nov 2008.

[12] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, Nov 2010.

[13] E. Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, USA, 1990.

[14] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: Can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, Oct. 2002.

[15] M. Wohlfart and H. Hauser. Story telling for presentation in volume visualization. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'07, pages 91–98. Eurographics Association, 2007.