Claude Code Handoff: 20K Canvas View Implementation

Project Context

Project: ProModel.ai Operational Dashboards - Warehouse Operations
Component: 20K ft Canvas View (3D Warehouse Visualization)
Repository: `rolandcedo/warehouse-data-viz-project` (public GitHub)
Phase: Interactive prototype development

Strategic Background

BigBear.ai is building operational dashboards for warehouse managers. The 20K Canvas View is the central visualization area where users explore their facility spatially. This replaces the current `CanvasPlaceholder.jsx` component.

Primary Persona: June - Warehouse Operations Manager

- Logs in at shift start asking: "How well will my facility perform vs plan?"
- Decision pattern: Visualize → Analyze → Optimize (sequential)
- Needs visibility into zone utilization, staffing, work content across time

Core UX Loop:

1. See facility overview (30K ft health score)
2. Identify problem areas via 20K canvas heatmaps
3. Click zone/entity to drill down (10K ft contextual panel)
4. Review recommendations, make decisions

Implementation Scope

What to Build

Primary Deliverable: 3D Isometric Warehouse View using Three.js

Supporting Components:

- View mode toolbar (top) — switches between 3D, 2D, Gantt views
- Heatmap controls (left panel) — toggles visualization overlays

- Selection system — zones, racks, equipment, people
- Time binding — colors/positions update with contextual time

Out of Scope (Existing)

- War Room views (DayShift, SwingShift, NightShift) — already implemented
- TimeScrubber — already implemented in AppLayout
- Contextual Sidepanel — already implemented, just needs data binding

---

Technical Specifications

Technology Stack

Concern	Technology	Notes
3D Rendering	**Three.js**	Scene, camera, renderer, raycasting
Camera	OrbitControls	RTS/city-builder style (Cities: Skylines inspiration)
Lighting	Ambient only	Flat, diagrammatic look — no shadows
Styling	Design system tokens	`C`, `sp`, `typography` from `designSystem.js`
State	React Context	Extend existing `TimeContext`, add `SelectionContext`
Icons	Lucide React	Consistent with rest of app

Performance Targets

- **60fps** on standard laptop
- **Mobile/tablet support** — low poly, no textures, grayscale base
- **~50 rack units**, ~20 people, ~10 equipment visible

Design System Reference

```javascript
// src/styles/designSystem.js

// Colors (use these for heatmaps)
C.success[500]  // #12B76A — Good (low utilization, fully staffed)
C.success[100]  // Light green background
C.warning[500]  // #F79009 — Moderate
C.warning[100]  // Light yellow background
C.error[500]    // #F04443 — Critical (high utilization, understaffed)
C.error[100]    // Light red background
C.neutral[200]  // #E5E5E5 — Off/neutral state
C.neutral[50]   // #FAFAFA — Canvas background
C.brand[500]    // #2F72FF — Selection highlight
C.purple[500]   // #7A5AF8 — Predictions

// Spacing
sp.xs   // 4px
sp.sm   // 8px
sp.md   // 16px
sp.lg   // 24px
sp.xl   // 32px
sp.xxl  // 48px
```

Heatmap Color Logic

Colors are **inverted based on metric type:**

Metric	Green (Good)	Yellow (Moderate)	Red (Critical)
Space Utilization	<70%	70-85%	>85%
Staffing Coverage	>100%	80-100%	<80%
Work Content	On track	Slight delay	Behind schedule
Foot Traffic	Normal flow	Congestion building	Gridlock

Component Architecture

```
src/
├── components/
│   └── Canvas/
│       ├── CanvasView.jsx          # Main container, view mode switching
│       ├── ThreeScene.jsx          # Three.js setup (renderer, camera, lights)
│       ├── Warehouse3D.jsx         # Warehouse geometry (floor, zones, racks)
│       ├── ZoneLayer.jsx           # Zone regions with selection/highlighting
│       ├── RackMesh.jsx            # Individual rack geometry
│       ├── EntitySprite.jsx        # 2D billboard sprites (people, equipment)
│       ├── HeatmapControls.jsx     # Left panel button group
│       ├── ViewModeToolbar.jsx     # Top toolbar icons
│       └── SelectionManager.jsx    # Raycasting + selection state
│
├── context/
│   ├── TimeContext.jsx             # Existing — contextual time state
│   └── SelectionContext.jsx        # NEW — selection state, breadcrumb path
│
├── data/
│   └── warehouseData.js            # NEW — centralized warehouse layout + entities
│
└── views/
    └── Executive.jsx               # Update to use CanvasView instead of CanvasPlaceholder
```

Data Architecture

Create: `src/data/warehouseData.js`

javascript

```
/**
 * Centralized warehouse data for 3D visualization
 * Single source of truth consumed by Canvas and Sidepanel
 */

export const WAREHOUSE_LAYOUT = {
  id: 'facility-001',
  name: 'Acme Distribution Center',
  dimensions: { width: 200, depth: 150 }, // meters (for Three.js scale)

  zones: [
    // Receiving
    {
      id: 'Z01',
      name: 'Inbound A',
      type: 'Receiving',
      color: 'success', // maps to C.success
      position: { x: -80, z: -60 },
      size: { width: 30, depth: 20 },
      metrics: {
        current: { utilization: 67, staffCount: 4, throughput: 120 },
        predicted: { utilization: 74, staffCount: 4, throughput: 145 } // at EOD
      },
      timeline: [
        { time: '06:00', utilization: 45 },
        { time: '08:00', utilization: 62 },
        { time: '10:00', utilization: 67 },
        { time: '12:00', utilization: 74 },
        { time: '14:00', utilization: 71 }
      ]
    },
    {
      id: 'Z02',
      name: 'Inbound B',
      type: 'Receiving',
      position: { x: -80, z: -35 },
      size: { width: 30, depth: 20 },
      metrics: {
        current: { utilization: 79, staffCount: 3, throughput: 98 },
        predicted: { utilization: 85, staffCount: 4, throughput: 130 }
      }
    },
    // QA
```

```
  {
    id: 'Z03',
    name: 'QA Station',
    type: 'Quality',
    position: { x: -45, z: -50 },
    size: { width: 20, depth: 30 },
    metrics: {
      current: { utilization: 75, staffCount: 2, throughput: 45 },
      predicted: { utilization: 80, staffCount: 2, throughput: 52 }
    }
  },
  // Reserve Storage (large)
  {
    id: 'Z04',
    name: 'Bulk Storage',
    type: 'Reserve',
    position: { x: -20, z: -40 },
    size: { width: 60, depth: 50 },
    metrics: {
      current: { utilization: 83, staffCount: 6, throughput: 200 },
      predicted: { utilization: 91, staffCount: 6, throughput: 180 }
    }
  },
  {
    id: 'Z05',
    name: 'Reserve 1',
    type: 'Reserve',
    position: { x: 45, z: -40 },
    size: { width: 35, depth: 25 },
    metrics: {
      current: { utilization: 70, staffCount: 3, throughput: 90 },
      predicted: { utilization: 68, staffCount: 3, throughput: 85 }
    }
  },
  {
    id: 'Z06',
    name: 'Reserve 2',
    type: 'Reserve',
    position: { x: 45, z: -10 },
    size: { width: 35, depth: 25 },
    metrics: {
      current: { utilization: 63, staffCount: 2, throughput: 75 },
      predicted: { utilization: 65, staffCount: 2, throughput: 80 }
    }
```

```
  },
  // Forward Pick (high activity)
  {
    id: 'Z07',
    name: 'Forward Pick A',
    type: 'Active',
    position: { x: -20, z: 15 },
    size: { width: 25, depth: 20 },
    metrics: {
      current: { utilization: 96, staffCount: 8, throughput: 450 },
      predicted: { utilization: 94, staffCount: 8, throughput: 420 }
    }
  },
  {
    id: 'Z08',
    name: 'Forward Pick B',
    type: 'Active',
    position: { x: 10, z: 15 },
    size: { width: 25, depth: 20 },
    metrics: {
      current: { utilization: 88, staffCount: 7, throughput: 380 },
      predicted: { utilization: 85, staffCount: 7, throughput: 360 }
    }
  },
  {
    id: 'Z09',
    name: 'Forward Pick C',
    type: 'Active',
    position: { x: 40, z: 15 },
    size: { width: 25, depth: 20 },
    metrics: {
      current: { utilization: 80, staffCount: 6, throughput: 320 },
      predicted: { utilization: 82, staffCount: 6, throughput: 340 }
    }
  },
  // Packing
  {
    id: 'Z10',
    name: 'Pack Station 1',
    type: 'Packing',
    position: { x: -20, z: 45 },
    size: { width: 20, depth: 15 },
    metrics: {
      current: { utilization: 75, staffCount: 4, throughput: 180 },
```

```
    predicted: { utilization: 78, staffCount: 4, throughput: 195 }
   }
  },
  {
   id: 'Z11',
   name: 'Pack Station 2',
   type: 'Packing',
   position: { x: 5, z: 45 },
   size: { width: 20, depth: 15 },
   metrics: {
    current: { utilization: 88, staffCount: 5, throughput: 210 },
    predicted: { utilization: 85, staffCount: 5, throughput: 200 }
   }
  },
  // Shipping
  {
   id: 'Z12',
   name: 'Shipping Dock A',
   type: 'Shipping',
   position: { x: -30, z: 65 },
   size: { width: 25, depth: 15 },
   metrics: {
    current: { utilization: 67, staffCount: 3, throughput: 150 },
    predicted: { utilization: 72, staffCount: 3, throughput: 165 }
   }
  },
  {
   id: 'Z13',
   name: 'Shipping Dock B',
   type: 'Shipping',
   position: { x: 0, z: 65 },
   size: { width: 25, depth: 15 },
   metrics: {
    current: { utilization: 83, staffCount: 4, throughput: 190 },
    predicted: { utilization: 88, staffCount: 4, throughput: 210 }
   }
  },
  // Special zones
  {
   id: 'Z14',
   name: 'Returns',
   type: 'Returns',
   position: { x: 70, z: -60 },
   size: { width: 20, depth: 25 },
```

```
      metrics: {
        current: { utilization: 47, staffCount: 2, throughput: 35 },
        predicted: { utilization: 52, staffCount: 2, throughput: 40 }
      }
    },
    {
      id: 'Z15',
      name: 'Hazmat',
      type: 'Hazmat',
      position: { x: 70, z: -30 },
      size: { width: 15, depth: 20 },
      metrics: {
        current: { utilization: 62, staffCount: 1, throughput: 15 },
        predicted: { utilization: 60, staffCount: 1, throughput: 14 }
      }
    },
    {
      id: 'Z16',
      name: 'Cold Storage',
      type: 'Cold',
      position: { x: 70, z: 20 },
      size: { width: 20, depth: 35 },
      metrics: {
        current: { utilization: 74, staffCount: 2, throughput: 60 },
        predicted: { utilization: 76, staffCount: 2, throughput: 65 }
      }
    }
  ]
};

// Rack data — positioned within zones
export const RACKS = [
  // Z04 Bulk Storage racks (main storage area)
  { id: 'R01', zoneId: 'Z04', position: { x: -35, z: -50 }, rotation: 0, slots: 48, occupied: 32 },
  { id: 'R02', zoneId: 'Z04', position: { x: -28, z: -50 }, rotation: 0, slots: 48, occupied: 45 },
  { id: 'R03', zoneId: 'Z04', position: { x: -21, z: -50 }, rotation: 0, slots: 48, occupied: 38 },
  { id: 'R04', zoneId: 'Z04', position: { x: -14, z: -50 }, rotation: 0, slots: 48, occupied: 41 },
  { id: 'R05', zoneId: 'Z04', position: { x: -7, z: -50 }, rotation: 0, slots: 48, occupied: 48 },
  { id: 'R06', zoneId: 'Z04', position: { x: 0, z: -50 }, rotation: 0, slots: 48, occupied: 44 },
  { id: 'R07', zoneId: 'Z04', position: { x: 7, z: -50 }, rotation: 0, slots: 48, occupied: 36 },
  { id: 'R08', zoneId: 'Z04', position: { x: 14, z: -50 }, rotation: 0, slots: 48, occupied: 29 },
  // ... continue for ~50 total racks across zones

  // Z07 Forward Pick A racks
```

```
  { id: 'R20', zoneId: 'Z07', position: { x: -28, z: 10 }, rotation: 0, slots: 32, occupied: 31 },
  { id: 'R21', zoneId: 'Z07', position: { x: -22, z: 10 }, rotation: 0, slots: 32, occupied: 30 },
  { id: 'R22', zoneId: 'Z07', position: { x: -16, z: 10 }, rotation: 0, slots: 32, occupied: 28 },

  // Add more racks to reach ~50 total...
];

// Entity data — people and equipment
export const ENTITIES = {
  people: [
    {
      id: 'P001',
      name: 'Maria Santos',
      role: 'Picker',
      zoneId: 'Z07',
      position: { x: -25, z: 12 },
      certifications: ['Forklift', 'Hazmat'],
      shift: 'Day',
      currentTask: 'Picking wave 2847'
    },
    {
      id: 'P002',
      name: 'James Wilson',
      role: 'Forklift Operator',
      zoneId: 'Z04',
      position: { x: -10, z: -45 },
      certifications: ['Forklift'],
      shift: 'Day',
      currentTask: 'Putaway'
    },
    // Add ~20 people total across zones
    { id: 'P003', name: 'Sarah Chen', role: 'Packer', zoneId: 'Z10', position: { x: -18, z: 47 }, shift: 'Day' },
    { id: 'P004', name: 'Mike Johnson', role: 'Unloader', zoneId: 'Z01', position: { x: -75, z: -58 }, shift: 'Day' },
    { id: 'P005', name: 'Lisa Park', role: 'QA Inspector', zoneId: 'Z03', position: { x: -42, z: -48 }, shift: 'Day' },
    { id: 'P006', name: 'Tom Bradley', role: 'Picker', zoneId: 'Z08', position: { x: 12, z: 18 }, shift: 'Day' },
    { id: 'P007', name: 'Ana Rodriguez', role: 'Picker', zoneId: 'Z09', position: { x: 42, z: 14 }, shift: 'Day' },
    { id: 'P008', name: 'Kevin Lee', role: 'Loader', zoneId: 'Z12', position: { x: -28, z: 68 }, shift: 'Day' },
    { id: 'P009', name: 'Emma Davis', role: 'Packer', zoneId: 'Z11', position: { x: 8, z: 46 }, shift: 'Day' },
    { id: 'P010', name: 'Chris Martin', role: 'Forklift Operator', zoneId: 'Z05', position: { x: 48, z: -38 }, shift: 'Day' },
    { id: 'P011', name: 'Rachel Kim', role: 'Picker', zoneId: 'Z07', position: { x: -20, z: 20 }, shift: 'Day' },
    { id: 'P012', name: 'David Brown', role: 'Unloader', zoneId: 'Z02', position: { x: -78, z: -32 }, shift: 'Day' },
    { id: 'P013', name: 'Jennifer Wu', role: 'Supervisor', zoneId: 'Z04', position: { x: 5, z: -35 }, shift: 'Day' },
    { id: 'P014', name: 'Robert Taylor', role: 'Returns Processor', zoneId: 'Z14', position: { x: 72, z: -55 }, shift: 'Day' },
    { id: 'P015', name: 'Amy Zhang', role: 'Cold Storage Tech', zoneId: 'Z16', position: { x: 72, z: 25 }, shift: 'Day' },
```

```
  ],

  equipment: [
    {
      id: 'E001',
      type: 'Forklift',
      name: 'FL-03',
      zoneId: 'Z04',
      position: { x: -5, z: -40 },
      status: 'Active',
      operator: 'P002'
    },
    {
      id: 'E002',
      type: 'Forklift',
      name: 'FL-07',
      zoneId: 'Z05',
      position: { x: 50, z: -35 },
      status: 'Active',
      operator: 'P010'
    },
    {
      id: 'E003',
      type: 'Pallet Jack',
      name: 'PJ-12',
      zoneId: 'Z07',
      position: { x: -22, z: 18 },
      status: 'Active',
      operator: 'P001'
    },
    { id: 'E004', type: 'Pallet Jack', name: 'PJ-15', zoneId: 'Z08', position: { x: 15, z: 12 }, status: 'Active' },
    { id: 'E005', type: 'Forklift', name: 'FL-11', zoneId: 'Z01', position: { x: -72, z: -55 }, status: 'Active' },
    { id: 'E006', type: 'Pallet Jack', name: 'PJ-22', zoneId: 'Z12', position: { x: -25, z: 70 }, status: 'Active' },
    { id: 'E007', type: 'Forklift', name: 'FL-15', zoneId: 'Z13', position: { x: 5, z: 68 }, status: 'Idle' },
    { id: 'E008', type: 'Conveyor', name: 'CV-Main', zoneId: 'Z10', position: { x: -10, z: 50 }, status: 'Running' },
  ]
};

// Heatmap configurations
export const HEATMAP_CONFIGS = {
  off: {
    id: 'off',
    label: 'Off',
    icon: 'Circle', // Lucide icon name
```

```
    getColor: () => '#E5E5E5' // neutral gray
  },
  spaceUtilization: {
    id: 'spaceUtilization',
    label: 'Space Utilization',
    icon: 'Box',
    metric: 'utilization',
    // Higher utilization = more red (bad for space)
    getColor: (value) => {
      if (value >= 90) return '#F04443'; // error
      if (value >= 80) return '#F79009'; // warning
      if (value >= 70) return '#FEC84B'; // warning light
      return '#12B76A'; // success
    }
  },
  workContent: {
    id: 'workContent',
    label: 'Work Content',
    icon: 'Package',
    metric: 'throughput',
    // Lower throughput vs target = red
    getColor: (value, target = 100) => {
      const pct = (value / target) * 100;
      if (pct < 70) return '#F04443';
      if (pct < 85) return '#F79009';
      if (pct < 95) return '#FEC84B';
      return '#12B76A';
    }
  },
  staffingWorkload: {
    id: 'staffingWorkload',
    label: 'Staffing Workload',
    icon: 'Users',
    metric: 'staffCount',
    // Lower staff coverage = red (inverted)
    getColor: (actual, planned = actual) => {
      const pct = (actual / planned) * 100;
      if (pct < 80) return '#F04443';
      if (pct < 90) return '#F79009';
      if (pct < 100) return '#FEC84B';
      return '#12B76A';
    }
  },
  footTraffic: {
```

```
    id: 'footTraffic',
    label: 'Foot Traffic',
    icon: 'Footprints',
    metric: 'traffic',
    // Higher traffic = more congestion = red
    getColor: (value) => {
      if (value >= 90) return '#F04443';
      if (value >= 75) return '#F79009';
      if (value >= 50) return '#FEC84B';
      return '#12B76A';
    }
  }
};

// View mode configurations
export const VIEW_MODES = {
  threeD: { id: 'threeD', label: '3D View', icon: 'Box' },
  twoD: { id: 'twoD', label: '2D View', icon: 'Square' },
  gantt: { id: 'gantt', label: 'Gantt View', icon: 'GanttChart' },
  warRoom: { id: 'warRoom', label: 'War Room', icon: 'Zap', navigateTo: 'warroom' }
};
```

Selection Context

Create: `src/context/SelectionContext.jsx`

javascript

```jsx
import React, { createContext, useContext, useState, useCallback } from 'react';

const SelectionContext = createContext();

export const useSelection = () => useContext(SelectionContext);

export const SelectionProvider = ({ children }) => {
  const [selectedPath, setSelectedPath] = useState([]); // Breadcrumb path
  const [selectedEntity, setSelectedEntity] = useState(null);
  const [hoveredEntity, setHoveredEntity] = useState(null);

  // Selection hierarchy: Facility > Zone > Rack/Equipment/Person
  const select = useCallback((type, id, data) => {
    setSelectedEntity({ type, id, data });

    // Build breadcrumb path
    const basePath = [{ type: 'facility', id: 'facility-001', label: 'Facility Overview' }];

    if (type === 'zone') {
      setSelectedPath([...basePath, { type: 'zone', id, label: data.name }]);
    } else if (type === 'rack') {
      const zone = data.zone;
      setSelectedPath([
        ...basePath,
        { type: 'zone', id: zone.id, label: zone.name },
        { type: 'rack', id, label: `Rack ${id}` }
      ]);
    } else if (type === 'person' || type === 'equipment') {
      const zone = data.zone;
      setSelectedPath([
        ...basePath,
        { type: 'zone', id: zone.id, label: zone.name },
        { type, id, label: data.name }
      ]);
    }
  }, []);

  const deselect = useCallback(() => {
    setSelectedEntity(null);
    setSelectedPath([{ type: 'facility', id: 'facility-001', label: 'Facility Overview' }]);
  }, []);

  const navigateBreadcrumb = useCallback((index) => {
```

```
    const item = selectedPath[index];
    if (index === 0) {
      deselect();
    } else {
      // Navigate to that level
      setSelectedPath(selectedPath.slice(0, index + 1));
      setSelectedEntity({ type: item.type, id: item.id });
    }
  }, [selectedPath, deselect]);

  return (
    <SelectionContext.Provider value={{
      selectedPath,
      selectedEntity,
      hoveredEntity,
      select,
      deselect,
      setHoveredEntity,
      navigateBreadcrumb
    }}>
      {children}
    </SelectionContext.Provider>
  );
};
```

Component Specifications

1. CanvasView.jsx (Main Container)

Responsibilities:

- Switches between view modes (3D, 2D, Gantt)
- Renders HeatmapControls on left
- Renders ViewModeToolbar at top
- Handles War Room navigation (exits to existing view)

Props:

```
javascript
```

```
{
  onNavigateToWarRoom: () => void,  // Navigate to War Room views
  onSelectEntity: (type, id, data) => void  // Update sidepanel
}
```

2. ThreeScene.jsx (Three.js Setup)

Responsibilities:

- Initialize Three.js renderer, scene, camera
- Setup OrbitControls (pan, zoom, rotate)
- Ambient lighting only
- Raycaster for mouse interaction
- Resize handling

Camera Defaults:

javascript

```
// Isometric overview position
camera.position.set(100, 80, 100);
camera.lookAt(0, 0, 0);

// OrbitControls settings
controls.enableDamping = true;
controls.dampingFactor = 0.05;
controls.maxPolarAngle = Math.PI / 2.2; // Prevent going below floor
controls.minDistance = 30;
controls.maxDistance = 250;
```

3. Warehouse3D.jsx (Geometry)

Responsibilities:

- Render floor plane with grid
- Render zone regions (flat colored planes)
- Render rack meshes
- Render entity sprites (people, equipment)

Zone Rendering:

javascript

```
// Zone as flat plane with border
const ZoneMesh = ({ zone, isSelected, isHovered, heatmapColor }) => {
  return (
    <group position={[zone.position.x, 0.1, zone.position.z]}>
      {/* Zone floor */}
      <mesh>
        <planeGeometry args={[zone.size.width, zone.size.depth]} />
        <meshBasicMaterial
          color={heatmapColor}
          transparent
          opacity={isSelected ? 1 : isHovered ? 0.9 : 0.7}
        />
      </mesh>

      {/* Selection outline */}
      {isSelected && (
        <lineSegments>
          <edgesGeometry args={[new BoxGeometry(zone.size.width, 0.5, zone.size.depth)]} />
          <lineBasicMaterial color="#2F72FF" linewidth={2} />
        </lineSegments>
      )}

      {/* Floating label */}
      <Html position={[0, 5, 0]} center>
        <div className="zone-label">{zone.name}</div>
      </Html>
    </group>
  );
};
```

4. EntitySprite.jsx (People/Equipment)

Responsibilities:

- Render 2D billboard sprites
- Hexagon shape for people
- Different shape for equipment
- Always face camera

javascript

```
// Using Three.js Sprite or @react-three/drei Html
const EntitySprite = ({ entity, type }) => {
  const color = type === 'person' ? '#2F72FF' : '#7A5AF8';

  return (
    <sprite position={[entity.position.x, 2, entity.position.z]}>
      <spriteMaterial>
        {/* Hexagon texture for people, square for equipment */}
      </spriteMaterial>
    </sprite>
  );
};
```

5. HeatmapControls.jsx (Left Panel)

Responsibilities:

- Vertical button group
- Radio-button behavior (one active at a time)
- Icons for each heatmap type

javascript

```
const HeatmapControls = ({ activeHeatmap, onChange }) => {
  const options = [
    { id: 'off', icon: Circle, label: 'Off' },
    { id: 'spaceUtilization', icon: Box, label: 'Space Utilization' },
    { id: 'workContent', icon: Package, label: 'Work Content' },
    { id: 'staffingWorkload', icon: Users, label: 'Staffing Workload' },
    { id: 'footTraffic', icon: Footprints, label: 'Foot Traffic' }
  ];

  return (
    <div style={{
      position: 'absolute',
      left: sp.md,
      top: '50%',
      transform: 'translateY(-50%)',
      display: 'flex',
      flexDirection: 'column',
      gap: sp.xs,
      background: 'white',
      padding: sp.sm,
      borderRadius: 8,
      boxShadow: '0 2px 8px rgba(0,0,0,0.1)'
    }}>
      {options.map(opt => (
        <button
          key={opt.id}
          onClick={() => onChange(opt.id)}
          style={{
            width: 40,
            height: 40,
            border: activeHeatmap === opt.id ? `2px solid ${C.brand[500]}` : `1px solid ${C.neutral[200]}`,
            borderRadius: 6,
            background: activeHeatmap === opt.id ? C.brand[50] : 'white',
            cursor: 'pointer',
            display: 'flex',
            alignItems: 'center',
            justifyContent: 'center'
          }}
          title={opt.label}
        >
          <opt.icon size={20} color={activeHeatmap === opt.id ? C.brand[500] : C.neutral[500]} />
        </button>
      ))}
```

```
    </div>
  );
};
```

6. ViewModeToolbar.jsx (Top Bar)

Responsibilities:

- Horizontal icon group
- Switch between 3D, 2D, Gantt
- War Room button navigates away

javascript

```
const ViewModeToolbar = ({ activeMode, onChange, onWarRoom }) => {
  const modes = [
    { id: 'warRoom', icon: Zap, label: 'War Room', action: onWarRoom },
    { id: 'twoD', icon: Square, label: '2D View' },
    { id: 'gantt', icon: GanttChart, label: 'Gantt View' },
    { id: 'threeD', icon: Box, label: '3D View' }
  ];

  return (
    <div style={{
      display: 'flex',
      gap: sp.xs,
      padding: sp.sm,
      background: 'white',
      borderRadius: 8
    }}>
      {modes.map(mode => (
        <button
          key={mode.id}
          onClick={() => mode.action ? mode.action() : onChange(mode.id)}
          style={{
            width: 44,
            height: 44,
            border: activeMode === mode.id ? `2px solid ${C.brand[500]}` : `1px solid ${C.neutral[200]}`,
            borderRadius: 8,
            background: activeMode === mode.id ? C.brand[500] : 'white',
            cursor: 'pointer'
          }}
          title={mode.label}
        >
          <mode.icon size={24} color={activeMode === mode.id ? 'white' : C.neutral[600]} />
        </button>
      ))}
    </div>
  );
};
```

Interaction Behaviors

Selection Flow

1. **Hover** → Show tooltip with name + key metric
2. **Click zone** →
    - Zone gets outline glow
    - Unselected zones dim (opacity 0.5)
    - Breadcrumb updates: "Facility > Zone Name"
    - Sidepanel shows zone details
3. **Click rack within zone** →
    - Rack highlights
    - Breadcrumb: "Facility > Zone > Rack ID"
    - Sidepanel shows rack details
4. **Click person/equipment** →
    - Entity highlights
    - Breadcrumb: "Facility > Zone > Entity Name"
    - Sidepanel shows entity details
5. **Click breadcrumb segment** → Navigate back to that level

Time Context Binding

When `contextualTime` changes in TimeContext:

1. Zone colors **transition smoothly** (300ms) to predicted values at that time
2. Entity positions **animate** to predicted locations
3. Tooltip metrics update to contextual values
4. "VIEWING: HH:MM" badge appears (already in TimeScrubber)

```javascript
```

```
// In Warehouse3D.jsx
const { contextualTime, interpolateValue } = useTimeContext();

// Get utilization at contextual time
const contextualUtilization = interpolateValue(zone.timeline, contextualTime);
const heatmapColor = getHeatmapColor(contextualUtilization);
```

Implementation Phases

Phase 1: Foundation (Start Here)

1. Create `warehouseData.js` with zone/rack/entity data
2. Create `SelectionContext.jsx`
3. Create `ThreeScene.jsx` with basic setup
4. Render floor plane and zone regions (colored boxes)
5. Implement zone selection + breadcrumb updates

Phase 2: Heatmaps

1. Create `HeatmapControls.jsx` button group
2. Implement color calculation based on active heatmap
3. Add smooth color transitions on heatmap change
4. Bind to TimeContext for predictive colors

Phase 3: Entities

1. Add rack meshes within zones
2. Add entity sprites (people, equipment)
3. Implement rack/entity selection
4. Add hover tooltips

Phase 4: Polish

1. Add floating zone labels

2. Implement opacity dimming on selection
3. Add selection outline glow
4. Animate entity position changes with time

Phase 5: Additional Views (Future)

1. 2D top-down view (camera snap)
2. Gantt view with task/event data
3. View mode transitions

Files to Modify

File	Action
`src/views/Executive.jsx`	Replace `CanvasPlaceholder` with `CanvasView`
`src/App.jsx`	Wrap with `SelectionProvider`, update sidepanel data binding
`src/components/ContextualSidepanel.jsx`	Consume selection context for dynamic content
`src/components/UI/Breadcrumb.jsx`	May need updates for selection-driven breadcrumbs

Dependencies to Install

```bash
npm install three @react-three/fiber @react-three/drei
```

- `three` — Core Three.js library
- `@react-three/fiber` — React renderer for Three.js
- `@react-three/drei` — Useful helpers (OrbitControls, Html, etc.)

Testing Checklist

- [ ] 3D scene renders with floor and zones
- [ ] OrbitControls work (pan, zoom, rotate)
- [ ] Zone colors reflect utilization heatmap
- [ ] Clicking zone updates breadcrumb
- [ ] Clicking zone updates sidepanel
- [ ] Heatmap toggle changes zone colors
- [ ] Time scrubber changes trigger color transitions
- [ ] Entity sprites render and are selectable
- [ ] Hover shows tooltip
- [ ] Selection dims unselected items
- [ ] Breadcrumb navigation works
- [ ] 60fps maintained

Reference Materials

- **Design Specs PDF:** `Warehouse_Design_Specs_v0_1.pdf` — UI compartments, mockups
- **Existing Components:** `ZonesView.jsx` — zone data structure reference
- **Design System:** `src/styles/designSystem.js` — colors, spacing, typography
- **TimeContext:** `src/context/TimeContext.jsx` — tri-temporal pattern

Questions for Implementation

If unclear during implementation, consider:

1. **Zone boundaries:** Zones should not overlap — adjust positions in `warehouseData.js` if needed
2. **Rack density:** Start sparse, add more racks if scene feels empty
3. **Entity movement:** For time-based position changes, interpolate between known positions
4. **Performance:** If fps drops, reduce entity count or simplify geometry first

---

Document generated: January 2026
For: Claude Code implementation of ProModel.ai 20K Canvas View